De-Anonymisierungsverfahren: Kategorisierung und deren Anwendung für Datenbankanfragen

Bachelorarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik
Lehrstuhl für Datenbank- und Informationssysteme

vorgelegt von: Johannes Goltz
Matrikelnummer: 213205061
geboren am: 03.07.1993 in Chemnitz
Erstgutachter: Prof. Dr. rer. nat. habil. Andreas Heuer
Zweitgutachter: Dr.-Ing. Thomas Mundt
Betreuer: M. Sc. Hannes Grunert
Abgabedatum: 06. Februar 2017

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AQuV	Answering Queries using Views	25
csv	Comma-separated values	45
DBMS	Datenbankmanagementsystem	19
DoS-Attacke	Denial-of-Service-Attacke	19
IDS	Intrusion-Detection-System	18
IPS	Intrusion-Prevention-System	18
JDBC	Java Database Connectivity	3
JSD	Jensen-Shannon-Divergenz	14
KLD	Kullback-Leibler-Divergenz	13
MCD	MiniCon Description	29
MySQL	My Structured Query Language	3
OSI-Layer	Open System Interconnect Layer	18
PArADISE	Privacy Aware Assistive Distributed Information System Environment	3
QI	Quasi-Identifikator	6
SQL	Structured Query Language	3

1. Einleitung

In der digitalisierten Gesellschaft werden den Menschen immer mehr Arbeiten von technischen Geräten abgenommen oder sie werden dabei von solchen unterstützt. Die elektronische Datenverarbeitung wird dadurch mit immer mehr persönlichen Daten konfrontiert. Diese Informationen sind für Konzerne sehr wichtig, um die Produkte noch gezielter auf die Bedürfnisse der Menschen zuschneiden zu können. Die Nutzer müssen sich inzwischen allerdings in Acht nehmen, da nahezu all diese Daten über sie gesammelt werden. Durch die Verknüpfung der unterschiedlichen Daten kann ein sehr genaues Profil einer Person erstellt werden. Zum Teil können sogar Entscheidungen vorausberechnet werden, bevor der Betroffene sich dieser bewusst wird.

1.1. Aktuelle Datenschutzproblematiken

In Amerika werden bereits erste Polizeirouten nach Algorithmen geplant, die abschätzen, wo es zu besonders vielen Straftaten kommen wird. Die Erkennung einer Schwangerschaft ist mit derartigen Algorithmen ebenfalls bereits möglich [Heu15]. Dieses Wissen kann auch genutzt werden, um entsprechenden Personen spezifische Werbung zusenden zu können. Allerdings muss bedacht werden was passiert, wenn die Algorithmen falsch liegen und unsere Handlungen missinterpretieren. Im Beispiel mit der Werbung scheint dies noch vergleichsweise unkritisch. Es könnte passieren, dass Bekannte von einer Schwangerschaft durch die gezielte Werbung erfahren, ein vergleichsweise unkritischer Fall. Was passiert aber, wenn die Polizeirouten falsch vorausgesagt werden oder Täter dies gezielt ausnutzen?

Aktuell zeigt sich ein weiteres Bedenken erregendes Beispiel beim Browser-Plugin „WOT - Sicher surfen“. Laut Recherchen des Norddeutschen Rundfunks sammelt dieses Plugin Nutzerdaten und leitet diese zu Auswertungszwecken anonymisiert weiter. Allerdings wurde die Anonymisierung nur sehr rudimentär umgesetzt, sodass eine De-Anonymisierung besonders leicht fällt [Eck16a]. Es reicht wohl zum Teil aus, die besuchten URLs nach Namen zu durchsuchen. Besonders kritisch ist es, da das Plugin eigentlich das Surfen im Netz sicherer gestalten soll, indem es Einschätzungen für die Sicherheit verschiedenartiger Seiten angibt. Laut den Recherchen war einfach nachvollziehbar, was Politiker oder auch Polizisten recherchieren und nach welcher Art von Informationen sie suchten. Auch ganze E-Mails, die online übersetzt wurden, konnten eingesehen werden und lieferten beispielsweise tiefe Einblicke in Ermittlungen [Eck16b]. Dieses Problem zeigt, wie wichtig es geworden ist, intensiv über Datenschutzkonzepte nachzudenken und personenbezogene Daten so zu anonymisieren, dass eine De-Anonymisierung nicht mehr möglich ist. Insbesondere bei Programmen oder Erweiterungen für Programme, die unsere Privatsphäre schützen sollen, erstaunt es umso mehr, wie fahrlässig mit Daten umgegangen wird.

Die Sammlung von Daten ist dabei zum Teil versteckt und zum Teil sehr offensichtlich. Wenn Produkte wie Sprachassistenten im Raum stehen, kann es auch für Besucher nicht sichtbar sein, dass Gespräche aufgezeichnet werden könnten. Es wird kontinuierlich alles Gesprochene im Raum ausgewertet und analysiert [Com16], um zu merken, wann eine Hilfestellung gewünscht ist. Auch wenn von den Anbietern versichert wird, dass nur Anfragen nach einem bestimmten Schlüsselwort an die Server weitergeleitet werden, muss sich der Kunde darauf verlassen. Ob die Analysen tatsächlich zum Großteil auf dem Gerät erfolgen, oder doch in Rechenzentren ausgeführt werden, ist hier die Frage. Sehr private Daten könnten schnell das eigene Heim verlassen und von möglicherweise Unbekannten analysiert werden. Dies geschieht im Zweifel nicht nur mit den privaten Daten, sondern mit allen Sprachaufzeichnungen der Anwesenden.

Deutlich unbewusster passiert derartiges aber sicherlich auf Mobiltelefonen. Hier werden alle möglichen Anwendungen genutzt. WhatsApp und Facebook gehören wohl zu den Bekanntesten. Beide Anwendungen gehören inzwischen zum Facebook-Konzern und WhatsApp hat seine Datenschutzrichtlinien gelockert. Seit dem werden ganze Telefonbücher, die für die Nutzung von WhatsApp wichtig sind, mit Facebook geteilt [Hur16]. Der Konzern hat damit eine unglaubliche Menge neuer Daten gewonnen und kann Beziehungen zwischen Menschen noch genauer „berechnen“ oder auch den Klarnamenzwang noch ganz anders durchsetzen. Apps wie Google Maps fragen Nutzer inzwischen auch von allein, ob Sie nicht Fotos von Objekten in der Umgebung dem Dienst zur Verfügung stellen möchten. Dies geschieht auch, wenn die App nicht geöffnet ist, allerdings die Standortortung am Telefon eingeschaltet ist.

Es sollte allerdings auch in Betracht gezogen werden, dass manche Menschen diese Dienste intensiv nutzen möchten. Daher ist es wünschenswert, dass der Nutzer selbst darüber entscheiden kann, inwieweit er seine Daten mit anderen teilen möchte. Sollte eine differenzierte Auswahl angeboten werden, muss auch an die Verständlichkeit gedacht werden. So muss es jedem Nutzer möglich sein, die angebotenen Auswahlen entsprechend bewerten zu können. Weiterhin ist zu differenzieren, ob die Daten innerhalb eines Konzerns weiterverarbeitet werden oder der Öffentlichkeit zur Verfügung gestellt werden. Veröffentlichte Daten sind besonders zu schützen, da es einem potentiellen Angreifer ohne Weiteres möglich ist, Zugang zu erlangen und diese Daten zu nutzen. Hier müssen also effiziente Datenschutzkonzepte greifen, die Personen im Einzelnen anonymisieren, aber trotzdem einen möglichst geringen Informationsverlust darstellen.

Der Verlust an Informationen kann dabei auch über den Fortbestand der Datenbank entscheiden. Je genauer die Daten und je besser diese zu verwerten sind, desto wichtiger wird die Datenbank für potentielle Kunden. Anders herum wird sie bei zu allgemeingültigen Datensätzen schnell uninteressant [Swe02].

1.2. Problemstellung

Diese Arbeit untersucht, inwieweit es möglich ist, die durch das Privacy Aware Assistive Distributed Information System Environment (PArADISE)[1]-Framework anonymisierten und bereitgestellten Daten wieder auf Personen beziehungsweise Handlungen, beziehbar zu machen – sprich die Anonymisierung umzukehren. Dabei soll vor allem auch der nötige Aufwand für ein derartiges Verfahren untersucht werden. Das Verfahren wird weiterhin nicht komplett automatisiert, sondern es soll eine Hilfestellung für einen Angreifenden gegeben werden. Diese wird für die korrekte Auswahl des Angriffs verwendet, um den nötigen Aufwand nach Möglichkeit auf ein Minimum zu reduzieren.

Als Datenbankmanagementsystem wird in dieser Arbeit auf My Structured Query Language (MySQL) gesetzt, wobei lediglich die Beispieldaten darin abgelegt sind. Die Anfragen sind in Structured Query Language (SQL) geschrieben. Auf diese Weise können die Ergebnisse der Arbeit sehr leicht mit anderen Datenbanksystemen genutzt werden. Die Anfragen sind daher von der späteren Verwendung unabhängig, da die Analysen auf den zurückgelieferten Java Database Connectivity (JDBC)-ResultSets erstellt werden. Bei der Anwendung mit Sensordaten und dem PArADISE-Framework werden ebenfalls ResultSets[2] zurückgegeben. Daher kann eine entsprechende Anwendung auch mit dieser Umgebung problemlos zusammenarbeiten.

Als Beispieldaten wurden keine Sensordaten des PArADISE-Frameworks genutzt, da diese sehr feingranular sind und Auswertungsanfragen, wie sie auch zur Sturzerkennung vorkommen, sehr kompliziert sind. Es wurde zur Veranschaulichung auf einfachere Datensätze und Anfragen gesetzt.

1.3. Leitfaden zur Arbeit

In der aktuellen Literatur wird eine Vielzahl unterschiedlicher Anonymisierungsverfahren beschrieben. Diese sind allerdings unter gewissen Voraussetzungen umkehrbar, beziehungsweise lassen sich mit Hintergrundwissen noch immer personenbezogene Daten extrahieren. In dieser Arbeit werden in Kapitel 3 Maße vorgestellt, mit denen sich der Grad der Anonymisierung genauer bestimmen lässt. Ebenso werden darin Verfahren beschrieben, die genutzt werden können, um die entsprechenden Maße zu erfüllen.

Kapitel 4 beschreibt Intrusion-Detection-Systeme, welche speziell für Datenbanken gedacht sind und auf der Schicht des Datenbankmanagementsystems arbeiten. Dabei wird zum Beispiel untersucht, welche Nutzer wie oft welche Daten anfragen und ob die gerade gestellte Anfrage aufgrund dieser Daten legitim ist. Ist es dem System möglich, dies vor Ausführung der Anfrage einzuschätzen, kann es auch als Intrusion-Prevention-System eingesetzt werden.

[1] Privacy Aware Assistive Distributed Information System Environment (PArADISE)
[2] Ein ResultSet stellt eine Tabelle dar, die das Ergebnis zu einer Datenbankanfrage enthält.

Im Weiteren wird untersucht, welche Angriffe existieren und welcher Aufwand beziehungsweise welche Voraussetzungen nötig sind, um diese durchzuführen. Genaueres dazu finden Sie in Kapitel 5. Dabei wird zwischen Verfahren unterschieden, die sich mit der Anfrage, oder nur mit den zurück gelieferten Daten beschäftigen.

Angriffe sind extrem schwierig allgemeingültig zu automatisieren. Daher soll im Zuge dieser Arbeit ein Programm entwickelt werden, welches einem potentiellen Angreifer die Möglichkeit gibt, die anonymisierten Daten zu analysieren. Aufgrund der Ergebnisse soll anschließend ein Angriff gewählt werden, der möglichst wenig zusätzliche Arbeit erfordert. Näheres ist in Kapitel 6 zu finden.

Abschließend gibt in Kapitel 7 eine Zusammenfassung über den Inhalt der Arbeit und die daraus gewonnenen Informationen. Es wird auch ein Blick auf weitere Forschungsthemen in diesem Bereich geworfen.

2. PArADISE

In dieser Arbeit soll insbesondere Bezug auf das PArADISE-Framework genommen werden. Dabei sind allerdings einige Punkte entsprechend angepasst worden. Näheres dazu in Abschnitt 2.3. Die Anonymisierung von personenbezogenen Daten gestaltet sich häufig schwieriger als gedacht. Unterschiedliche Anfragen zur Auswertung der Ergebnisse brauchen zudem verschiedenartige Arten der Anonymisierung, um effizient arbeiten zu können. PArADISE kann den Entwickler oder auch Forscher hierbei unterstützen [Gru16].

2.1. Ausgangssituation

An der Universität Rostock wird interdisziplinär an Assistenzsystemen geforscht, etwa für die Erkennung von Stürzen in Wohnungen. Das PArADISE-Framework wird besonders für diesen Anwendungsfall entwickelt und genutzt. Neben der Sensorik wird auch die Auswertung und entsprechende Datenverarbeitung untersucht.

PArADISE dient dabei als Treiber für die Verarbeitung der entsprechenden SQL-Anfragen. Dabei wurde ein Schichtsystem umgesetzt, wobei sich die Schichten nach der Leistungsfähigkeit der entsprechend zugehörigen Geräte einteilen lassen. Beispielhaft ist dies in Abbildung 2.1 dargestellt.

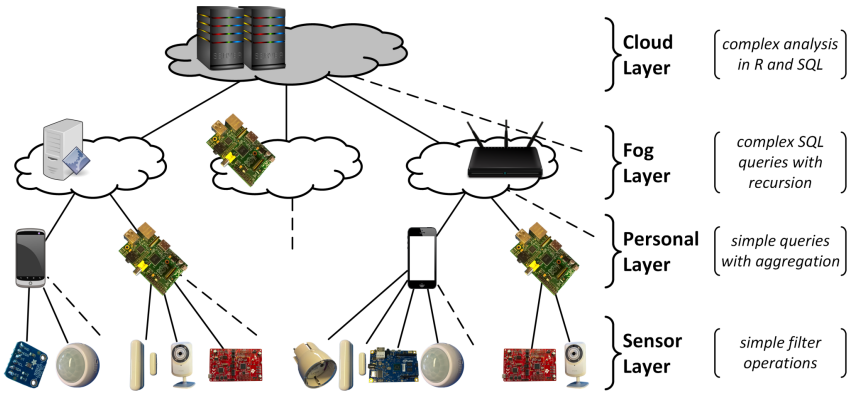

Abbildung 2.1.: Schichtaufbau des PArADISE-Frameworks [GKB+16]

Auf der niedrigsten Schicht (Sensor Layer) liegen die Sensoren. Die Menge dieser kann sich zur Laufzeit dynamisch anpassen, sobald neue Sensoren hinzukommen, oder andere wegfallen. Es folgen einzelne Geräte wie Home Media Center, intelligente Fernseher oder auch Mobiltelefone (Personal Layer), ein

lokaler Server in der Wohnung (Fog Layer) und ein Cloud-Server (Cloud Layer) auf der höchsten Ebene. Es wird versucht, die Anfragen so früh wie möglich auszuführen, um so wenige Daten wie möglich an höhere Schichten weiter reichen zu müssen. Dadurch wird erreicht, dass möglichst wenige Daten den persönlichen Einflussbereich verlassen und Dritten mitgeteilt werden. Sollte die Ausführung aufgrund von zu schwacher Hardware oder zu komplexen Anfragen nicht möglich sein, so wird die Anfrage aufgeteilt. Nur der Teil, welcher nicht ausführbar ist, wird mit den dafür benötigten Daten an die höhere Schicht weitergeleitet. Bevor Daten an die Cloud-Sicht weitergeleitet werden, wird allerdings überprüft, ob sie die Ansprüche an die Anonymisierung erfüllen. Sollte dies nicht der Fall sein, wird dies mit einem entsprechenden Algorithmus zur weiteren Anonymisierung korrigiert. Dadurch wird sichergestellt, dass nur die absolut notwendigen Daten übertragen werden. Dies beugt einem Missbrauch der Daten zur Extraktion weiterer Informationen vor und verbessert damit die Privatsphäre des Nutzers.

2.2. Aufbau

Das Framework ist modular aufgebaut. Der Hauptbestandteil wird durch den Anfrageprozessor dargestellt. Durch PArADISE ist es auch möglich, Anfragen im Präprozessor umzuschreiben, beziehungsweise diese aufzuteilen. Es kann daher auch ein Schichtkonzept entwickelt werden. Dies bietet aus Sicht des Datenschutzes den Vorteil, dass Anfragen frühestmöglich ausgewertet werden können und so nur die unbedingt notwendigen Daten an höhere Schichten weitergegeben werden. Damit bleiben die Daten so nah wie möglich an der Quelle und verlassen im Idealfall nicht den Wirkungsbereich des Nutzers.

Sollten die Anfragen allerdings nicht am aktuellen Knoten ausgeführt werden können, so kann ein entsprechender Algorithmus zur Anonymisierung genutzt werden, um die Daten im Postprozessor entsprechend für die Weitergabe vorzubereiten. Dabei sind verschiedene Verfahren implementiert, sodass sowohl spaltenbasierte, als auch zeilenbasierte Auswertungen durchgeführt werden können. Um all dies zu bewerkstelligen, werden zusätzliche Informationen im Anfrageprozessor benötigt. Dazu zählen die Datenschutzeinstellungen des Nutzers, die Leistungsfähigkeit des Rechners oder auch des aktuellen Knotens und der Informationsbedarf des Systems [Gru16]. Verdeutlicht ist dies in Abbildung 2.2. Der Anfrageprozessor kann dabei mehrfach hintereinander geschaltet werden und auf jeder Schicht eigenständig eingesetzt werden.

Während des Betriebs wird zusätzlich geprüft, ob in Anfragen ein Quasi-Identifikator (QI) (siehe Definition 1) enthalten ist und ob daher entsprechend eingegriffen werden muss. Ein Eingriff würde dann die Anonymisierung der Daten vornehmen, so dass die entsprechend übergebenen Richtlinien erfüllt werden. Es ist auch möglich, die angewandten Techniken aus den Metadaten nach der Anonymisierung auszulesen, um so das Provenance Management[1] zu ermöglichen.

[1] Beim Provenance Management geht es darum, die Prozesse, welche die Daten verändern, zu erfassen, um eine Nachvollziehbarkeit über die vorgenommenen Änderungen zu gewährleisten. Dies ist besonders bei einer weiteren Verarbeitung wichtig.

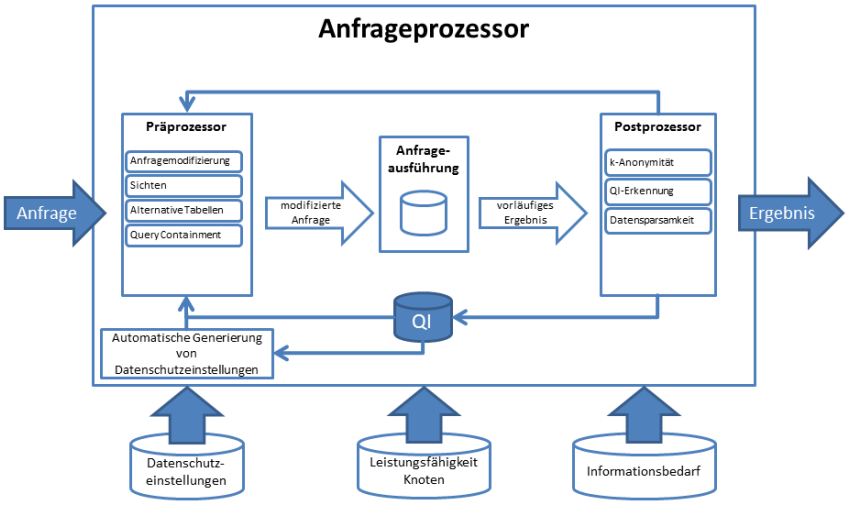

Abbildung 2.2.: Anfrageprozessor des PArADISE-Frameworks [GH15]

2.3. Annahmen und Anpassungen in dieser Arbeit

Um den entwickelten Treiber und die Anonymisierungstechniken entsprechend überprüfen zu können, wird hier nicht auf Basis komplexen Anfragen mit den Sensordaten als Grundlage gearbeitet, sondern es werden vereinfachte SQL-Anfragen auf dem *adult*- [UCI96] und dem *employee*-Datensatz [MyS] ausgeführt. Zudem trifft hier die Anonymisierung der Daten einen etwas anderen Bereich als dies „klassisch“ der Fall wäre. Da die Daten immer aus einer Wohnung stammen sind sie entsprechend einfach personenbeziehbar. Im Vordergrund steht bei diesem Projekt allerdings die Anonymisierung der Handlungen. Es soll möglichst nur das benötigte Wissen aus den bereitgestellten Daten ableitbar sein und nicht mehr. Dafür werden die entsprechenden Anonymisierungskonzepte hier zur Anwendung gebracht. Sollte keine Anonymisierung vorgenommen werden, wären eventuell auch andere Verhaltensweisen als Stürze erkennbar.

In der hier verfassten Arbeit wird allerdings wieder die „klassische“ Anonymisierung der Daten betrachtet, da dies entsprechend zu den Beispieldaten passt. In Kapitel 6 wird eine potentielle Durchführung eines Angriffs besprochen. In diesem Kapitel wird ebenfalls die neu entwickelte Hilfestellung für Angriffe vorgestellt. Dabei wurde ebenfalls nicht auf den modifizierten Treiber des PArADISE-Frameworks zurückgegriffen, sondern es wurde ein Standard-MySQL-Treiber verwendet. Diese Treiber sind allerdings direkt austauschbar, sodass die entwickelte Software auch problemlos mit dem modifizierten Treiber und den Daten zur Sturzerkennung arbeiten kann.

3. Anonymisierungsverfahren und -maße

In diesem Abschnitt sollen einige bestehende Anonymisierungsverfahren und -maße vorgestellt werden, um einen Überblick über den aktuellen Stand zu erhalten. Besonders wichtig sind für alle Konzepte die sogenannten Quasi-Identifikatoren. In der Regel gibt es in jeder Tabelle einer Datenbank einen eindeutigen Schlüssel, um jedes Tupel der Tabelle explizit ansprechen zu können. In einer anonymisierten Version darf ein derartiger Schlüssel selbstverständlich nicht vorkommen. Quasi-Identifikatoren können allerdings weiterhin enthalten sein und müssen im Zweifel weiter anonymisiert werden.

Definition 1. *Ein Quasi-Identifikator Q_T ist eine endliche Menge von Attributen $\{A_i, ..., A_j\}$ einer Tabelle T mit einer endlichen Menge von Attributen $\{A_1, A_2, ..., A_n\}$. Hierbei gilt $\{A_i, ..., A_j\} \subseteq \{A_1, A_2, ..., A_n\}$. Mit Hilfe des Quasi-Identifikators ist es möglich, mindestens ein Tupel der Tabelle T eindeutig bestimmen zu können [Hau07]. Eine Menge von Tupeln t einer Tabelle T, welche bezüglich des Quasi-Identifikators Q_T nicht unterscheidbar sind, wird als q^*-Block bezeichnet.*

Im Weiteren wird auch die Bedeutung der sensitiven Attribute wichtig. Diese stellen Attribute außerhalb der Menge der Quasi-Identifikatoren dar. Ein Attribut wird als sensitiv beschrieben, sobald es einer Person schaden kann oder auch deren Privatsphäre verletzt, wenn der entsprechende Wert dieser Person zugeordnet werden kann [Hau07].

Quasi-Identifikatoren können, wie in Definition 2 gezeigt, noch genauer bestimmt werden. In dieser Arbeit wird immer von der α-distinct-Quasi-Identifikatoren ausgegangen.

Definition 2. *Quasi-Identifikatoren lassen sich nach folgenden Klassen einteilen (nach [MX07]):*

1. *Ein α-distinct-Quasi-Identifikator ist eine Teilmenge von Attributen, welche zum Schlüssel der Tabelle wird, die übrig bleibt, wenn maximal ein Anteil von $1 - \alpha$ Tupeln gestrichen wird.*
2. *Eine Teilmenge von Attributen separiert ein Paar von Tupeln x und y, wenn x und y unterschiedliche Werte in wenigstens einem der Attribute der Teilmenge aufweisen. Ein α-separierender-Quasi-Identifikator ist eine Teilmenge an Attributen, welche mindestens einen α-Anteil aller Tupelpaare separieren.*

Um dies nochmal zu verdeutlichen soll das Beispiel aus Tabelle 3.1 dienen.

3.1. Generalisierung und Unterdrückung

Anonymisierung kann durch verschiedene Verfahren erreicht werden, zum Beispiel Generalisierung und Unterdrückung. Grundsätzlich ist die Generalisierung in Hinsicht auf Tabellen ein spaltenorientiertes

Zeile	age
1	39
2	31
3	42
4	30
5	30

Tabelle 3.1.: ResultSet folgender Anfrage: **SELECT** age **FROM** adult **WHERE** capitalGain > 0 **LIMIT** 5;, Das Attribut *age* ist ein 0,8-distinct-Quasi-Identifikator (4 unterschiedliche Werte auf 5 Tupeln gesamt) und ein 0,8125-separierender-Quasi-Identifikator (13 von 16 Wertepaaren aus dem Alter können unterschieden werden.)

Verfahren, das heißt hiermit lässt sich der Inhalt von einzelnen Attributen allgemeiner ausdrücken. Die Unterdrückung hingegen arbeitet zeilenorientiert. Dabei werden einzelne Tupel unterdrückt. Dies ist dann sinnvoll, wenn ansonsten eine Vielzahl von weiteren Generalisierungen nötig wäre [Sam98].

Bei der Generalisierung wird von sogenannten Domänen ausgegangen. Dabei bildet jedes Attribut eine Domäne. Der Wert in der Ausgangstabelle bildet die Grunddomäne. Ausgehend von dieser werden die Werte stufenweise in weiteren Domänen verallgemeinert. Das Alter könnte so beispielsweise von expliziten Zahlen zu Gruppen von je fünf Werten zusammengefasst werden, dann Zehner, dann Zwanziger, je nachdem wie weit dies gewünscht ist. Auf diese Art und Weise ist es möglich, den Informationsgehalt in der Tabelle abzuschwächen und den Grad der Anonymisierung zu erhöhen [Sam98].

Das Verfahren der Unterdrückung arbeitet, wie bereits erwähnt, auf Zeilenebene. Dabei werden die Werte nicht verallgemeinert, sondern komplette Tupel in Gänze unterdrückt. Sollten einzelne Tupel eine starke Generalisierung nötig machen, um einen gewissen Grad an Anonymisierung zu erhalten, können diese einfach gestrichen werden. Damit wird der Informationsverlust bei gleichbleibender Anonymisierung verringert und eine bessere Auswertung aufgrund der Daten kann ermöglicht werden. Um möglichst gute Ergebnisse zu erreichen, sollten beide Verfahren kombiniert werden. Um die Nutzung der Unterdrückung zu limitieren, kann es sinnvoll sein, einen Maximalwert für die Anzahl zu unterdrückter Tupel anzugeben. Damit wird sichergestellt, dass trotzdem noch ein repräsentatives Ergebnis vorliegt [Sam98].

3.2. k-Anonymität

Dieses Maß stellt vergleichsweise geringe Ansprüche bezüglich der Quasi-Identifikatoren an die Anonymisierung. Je größer der Wert k gewählt wird, desto besser die Anonymisierung. k-Anonymität gewährleistet dabei, dass eine Person von $k-1$ anderen Personen bezüglich der Quasi-Identifikatoren nicht unterscheidbar ist.

Definition 3. *Sei T eine Tabelle mit persönlichen Daten. $\{A_1, \ldots, A_n\}$ sei die endliche Menge an Attributen in T. Q_T sei der bekannte Quasi-Identifikator von T. Q_T setzt sich aus den Attributen $\{A_i, \ldots, A_k\} \subseteq \{A_1, \ldots, A_n\}$ zusammen. Eine Menge von Tupeln $\{t_1, \ldots, t_l\}$ aus Tabelle T ist k-anonym,*

wenn in T k Tupel existieren, für die gilt: $t_1[A_i, \ldots, A_k] = \ldots = t_l[A_i, \ldots, A_k]$. *Sobald für alle Wertkombinationen von* $Q_T = \{A_i, \ldots, A_k\}$ *jeweils mindestens k Tupel existieren, erfüllt die Tabelle k-Anonymität [Hau07].*

Als Beispiel dienen dazu die Tabellen 3.2 und 3.3. In Tabelle 3.2 sind die Rohdaten als originaler Auszug des Adult-Datensatzes [UCI96] für die folgende Anfrage enthalten.

```
SELECT age, education, maritalStatus, race, sex, capitalGain
FROM adult
WHERE capitalGain > 0
LIMIT 10;
```

Tabelle 3.3 enthält die gleichen Daten, allerdings per Generalisierung verallgemeinert. Dadurch erfüllt die Tabelle nun k-Anonymität für $k = 2$. Mittels Unterdrückung ist es möglich das Alter beispielsweise genauer einzugrenzen, wenn dafür Zeile 9 in Tabelle 3.2 gestrichen wird. Die einzelnen q^*-Blöcke wurden farblich hervorgehoben.

Zeile	age	education	maritalStatus	sex	race	capitalGain
1	39	Bachelors	Never-married	White	Male	2174
2	31	Master	Never-married	White	Female	14084
3	42	Bachelors	Married-civ-spouse	White	Male	5178
4	30	HS-grad	Married-civ-spouse	White	Male	5013
5	30	Bachelors	Married-civ-spouse	White	Male	2407
6	44	HS-grad	Divorced	White	Female	14344
7	44	Bachelors	Married-civ-spouse	White	Male	15024
8	32	HS-grad	Married-civ-spouse	White	Male	7688
9	17	10th	Never-married	White	Female	34095
10	28	Some-college	Married-civ-spouse	White	Female	4064

Tabelle 3.2.: ResultSet folgender Anfrage: **SELECT** age, education, maritalStatus, race, sex, capitalGain **FROM** adult **WHERE** capitalGain>0 **LIMIT** 10;

Zeile	age	education	maritalStatus	race	sex	capitalGain
1	31-60	10th or better	married or not	White	Male	2174
2	31-60	10th or better	married or not	White	Female	14084
3	31-60	10th or better	married or not	White	Male	5178
4	0-30	10th or better	married or not	White	Male	5013
5	0-30	10th or better	married or not	White	Male	2407
6	31-60	10th or better	married or not	White	Female	14344
7	31-60	10th or better	married or not	White	Male	15024
8	31-60	10th or better	married or not	White	Male	7688
9	0-30	10th or better	married or not	White	Female	34095
10	0-30	10th or better	married or not	White	Female	4064

Tabelle 3.3.: Mittels Generalisierung anonymisierte Tabelle (aus Tabelle 3.2) die k-Anonymität für $k = 2$ erfüllt. (QI: {*age*, *education*, *maritalStatus*, *race*, *sex*})

3.3. l-Diversität

Ein großes Problem von k-Anonymität ist eine geringe Vielfältigkeit in Hinblick auf das sensitive Attribut. Angenommen eine anonymisierte Tabelle erfüllt die k-Anonymität für $k = 2$. Ein Angreifer hat eine Gruppe ermittelt, in der die Person seines Interesses liegt. Für das sensitive Attribut ist allerdings bei beiden Tupeln der Gruppe der gleiche Wert vorhanden. Dies ist möglich, da es laut k-Anonymität keine Vielfalt für das sensitive Attribut geben muss. Angreifern mit Hintergrundwissen ist es somit möglich, trotzdem Rückschlüsse auf einzelne Individuen zu ziehen. Dies soll durch das Maß der l-Diversität vermieden werden [MKGV07].

Definition 4. *Es sei eine k-anonymisierte Tabelle T^* von T gegeben. Ein q^*-Block ist l-divers, wenn mindestens l unterschiedliche Werte für das sensitive Attribut im q^*-Block vorkommen. Eine Tabelle kann als l-divers bezeichnet werden, sobald dies für jeden q^*-Block der Tabelle gilt [Hau07].*

Zeile	Geburtsdatum	Postleitzahl	Ort	Geschlecht	Diagnose
1	05.08.1985	18055	Rostock	männlich	Migräne
2	09.12.1965	18059	Rostock	männlich	Mittelohrentzündung
3	02.06.1955	18057	Rostock	männlich	Darmkrebs
4	13.08.1968	18055	Rostock	männlich	Schlaganfall
5	19.05.1962	18069	Rostock	weiblich	Inkontinenz
6	18.07.1994	18181	Graal-Müritz	weiblich	Lungenentzündung
7	07.01.1989	18069	Rostock	weiblich	gebrochenes Bein
8	04.06.1973	18059	Rostock	männlich	Diabetes
9	28.03.1969	18181	Graal-Müritz	weiblich	Nierenversagen
10	14.09.1990	18057	Rostock	männlich	Hodenkrebs

Tabelle 3.4.: Beispieldatensatz (QI: {*Geburtsdatum*, *Postleitzahl*, *Ort*, *Geschlecht*})

Zeile	Geburtsdatum	Postleitzahl	Ort	Geschlecht	Diagnose
1	1950-1999	18055	Rostock	männlich	Migräne
2	1950-1999	18059	Rostock	männlich	Mittelohrentzündung
3	1950-1999	18057	Rostock	männlich	Darmkrebs
4	1950-1999	18055	Rostock	männlich	Schlaganfall
5	1950-1999	18069	Rostock	weiblich	Inkontinenz
6	1950-1999	18181	Graal-Müritz	weiblich	Lungenentzündung
7	1950-1999	18069	Rostock	weiblich	gebrochenes Bein
8	1950-1999	18059	Rostock	männlich	Diabetes
9	1950-1999	18181	Graal-Müritz	weiblich	Nierenversagen
10	1950-1999	18057	Rostock	männlich	Hodenkrebs

Tabelle 3.5.: Mittels Generalisierung anonymisierte Tabelle (aus Tabelle 3.4) die k-Anonymität für $k = 2$ und l-Diversität für $l = 2$ erfüllt. (QI: {*Geburtsdatum*, *Postleitzahl*, *Ort*, *Geschlecht*})

In Tabelle 3.5 ist ersichtlich, dass für jeden q^*-Block zwei unterschiedliche Diagnosen bestehen. Damit ist l-Diversität für $l = 2$ erfüllt. Als Voraussetzung müssen selbstverständlich zuerst entsprechende q^*-Blöcke

gebildet werden. Dies passiert mittels Generalisierung des Geburtsdatums auf einen Bereich von 50 Jahren. Eine engere Eingrenzung ist bei diesem Datenbestand so nicht umzusetzen, oder führt auf der anderen Seite zu einem großen Verlust an Informationen in den weiteren Spalten.

Ein Problem der l-Diversität ist, dass nicht auf die unterschiedliche Semantik und die Verteilung der Werte des sensitiven Attributes geachtet wird. So ist es beispielsweise einem Angreifer in Tabelle 3.5 möglich, sein „Opfer“ in den q^*-Block aus Zeile drei und zehn (türkis) einzuordnen, da er die Postleitzahl kennt. Auf der Seite der Diagnose erfährt er somit, dass das „Opfer“ entweder an Hoden- oder an Darmkrebs leidet. Diese Information allein kann aber schon ausreichend sein, um sie zum Nachteil des Betroffenen zu nutzen. Ähnlich ist dies in der Tabelle zur k-Anonymität denkbar (siehe Tabelle 3.3). Hier reicht dem Angreifer vielleicht schon das Wissen, ob der Wert für „capitalGain“ unter oder über 10000 liegt. Beim roten und blauen q^*-Block wäre auch dies schon eindeutig und es wäre keine ausreichende Anonymität gewährleistet.

3.4. t-Closeness

k-Anonymität und l-Diversität können jeweils keine absolute Anonymisierung gewährleisten. Mit Hilfe von t-Closeness ist dies möglich, allerdings führt dieses Maß zu einem hohen Informationsverlust bei sehr strikter Anwendung [Dom08]. Bei diesem Maß wird neben der Vielfalt des sensitiven Attributs auch auf die Verteilung der Werte geachtet. Dabei muss jeder q^*-Block den anderen q^*-Blöcken hinsichtlich der Verteilung des sensitiven Attributes möglichst gleich sein. Ein potentieller Angreifer kann dadurch, selbst wenn er den q^*-Block identifiziert, in dem sein „Opfer“ enthalten ist, nicht mehr über es lernen, als wenn es in einem anderen q^*-Block stehen würde. Bei diesem Maß gilt daher auch je kleiner t, desto höher der Grad der Anonymisierung [Hau07].

Definition 5. *Ein q^*-Block erfüllt t-Closeness, sobald der Abstand seiner sensitiven Attribute einen Grenzwert t zu denen der gesamten Tabelle nicht überschreitet. Eine Relation erfüllt demnach t-Closeness, sobald jeder ihrer q^*-Blöcke dies erfüllt [Hau07].*

Zeile	Geburtsdatum	Ort	Geschlecht	Diagnose
1	05.08.1985	Graal-Müritz	weiblich	Virus-Infektion
2	09.12.1965	Rostock	männlich	Virus-Infektion
3	02.06.1955	Rostock	männlich	Krebs
4	13.08.1990	Graal-Müritz	weiblich	Organversagen
5	19.05.1962	Rostock	männlich	Organversagen
6	18.07.1994	Graal-Müritz	weiblich	Krebs
7	28.03.1975	Graal-Müritz	weiblich	Organversagen
8	14.09.1968	Rostock	männlich	Krebs

Tabelle 3.6.: Beispieldatensatz (QI: {*Geburtsdatum*, *Ort*, *Geschlecht*})

Zeile	Geburtsdatum	Ort	Geschlecht	Diagnose
1	1975-1999	Graal-Müritz	weiblich	Virus-Infektion
2	1950-1974	Rostock	männlich	Virus-Infektion
3	1950-1974	Rostock	männlich	Krebs
4	1975-1999	Graal-Müritz	weiblich	Organversagen
5	1950-1974	Rostock	männlich	Organversagen
6	1975-1999	Graal-Müritz	weiblich	Krebs
7	1975-1999	Graal-Müritz	weiblich	Organversagen
8	1950-1974	Rostock	männlich	Krebs

Tabelle 3.7.: Anonymisierter Datensatz aus Tabelle 3.6, der k-Anonymität für $k = 4$, l-Diversität für $l = 3$ und t-Closeness für $t \approx 0,0637$ erfüllt (Berechnung mit Hilfe der Kullback-Leibler-Divergenz).

	Virus-Infektion	Krebs	Organversagen
p_T	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{3}{8}$
$p_{T'}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$
$p_{T''}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

Tabelle 3.8.: Wahrscheinlichkeitsverteilungen (für Tabelle 3.7) für die einzelnen Werte des sensitiven Attributs (Diagnose) in der gesamten Tabelle (p_T), dem roten q^*-Block ($p_{T'}$) und dem türkisen q^*-Block ($p_{T''}$).

t-Closeness mit Kullback-Leibler-Divergenz

Eine Schwierigkeit bildet hier die Messung der Distanz der unterschiedlichen Werte. Während die Distanzmessung der Verteilungen numerischer Werte noch vergleichsweise leicht fällt, wird es vor allem bei kategorischen Attributen schwieriger. Hier bietet sich vor allem die Kullback-Leibler-Divergenz (KLD) an [Sha10].

Definition 6. *Gegeben seien zwei Wahrscheinlichkeitsverteilungen $p = (p_1, ..., p_N)$ und $q = (q_1, ..., q_N)$. Die KLD zwischen p und q wird wie folgt berechnet [Sha10]:*

$$KL(p||q) = \sum_{i=1}^{N} p_i \cdot log_2 \frac{p_i}{q_i}$$

Tabelle 3.6 besitzt als QI die Attribute *Geburtsdatum*, *Ort* und *Geschlecht*. Das sensitive Attribut sei *Diagnose*. Um t-Closeness zu erfüllen, müssen zuerst q^*-Blöcke gebildet werden, für die entsprechend der Wert berechnet werden kann. Tabelle 3.7 erfüllt diese Bedingung und enthält zwei q^*-Blöcke mit jeweils vier Tupeln. Damit ist k-Anonymität für $k = 4$ gewährleistet. Da in jedem q^*-Block mindestens drei unterschiedliche Diagnosen vorkommen, ist ebenfalls l-Diversität für $l = 3$ erfüllt. Zur Berechnung von t dient Tabelle 3.8 als Hilfestellung. Hier werden die Wahrscheinlichkeiten für das Auftreten der einzelnen konkreten Werte des sensitiven Attributs in Bezug auf die gesamte Tabelle und in Bezug auf die einzelnen q^*-Blöcke gezeigt. Die Distanz wird nach Definition 6 berechnet. Zu beachten ist, dass der Wert für jeden

q^*-Block berechnet werden muss und anschließend das Maximum den Wert für t-Closeness darstellt. (Berechnung siehe Abbildung 3.1)

$$
\begin{aligned}
KL(T||T') &= \sum_{i=1}^{3} T_i \cdot log_2 \frac{T_i}{T'_i} \\
KL(T||T') &= \frac{1}{4} \cdot log_2 \frac{\frac{1}{4}}{\frac{1}{4}} + \frac{3}{8} \cdot log_2 \frac{\frac{3}{8}}{\frac{1}{4}} + \frac{3}{8} \cdot log_2 \frac{\frac{3}{8}}{\frac{1}{2}} \\
KL(T||T') &\approx 0,0637 \\
KL(T||T'') &= \sum_{i=1}^{3} T_i \cdot log_2 \frac{T_i}{T''_i} \\
KL(T||T'') &= \frac{1}{4} \cdot log_2 \frac{\frac{1}{4}}{\frac{1}{4}} + \frac{1}{2} \cdot log_2 \frac{\frac{3}{8}}{\frac{1}{2}} + \frac{1}{4} \cdot log_2 \frac{\frac{3}{8}}{\frac{1}{4}} \\
KL(T||T'') &\approx 0,0637 \\
max(KL(T||T'), KL(T||T'')) &= KL(T||T') = KL(T||T'') \approx 0,0637
\end{aligned}
\tag{3.1}
$$

Abbildung 3.1.: Berechnung des Wertes für t-Closeness (für Tabelle 3.7) mittels KLD

t-Closeness mit Jensen-Shannon-Divergenz

Der Wert t für t-Closeness kann anstelle mit der KLD auch mit der Jensen-Shannon-Divergenz (JSD) berechnet werden. Diese stellt eine angepasste Version der KLD und löst das Problem, dass ein Divisor den Wert 0 annehmen könnte. Sollte ein Attributwert des sensitiven Attributes in einem q^*-Block nicht auftauchen, so nimmt der Divisor den Wert 0 an. Dies macht eine Berechnung mit der KLD unmöglich. Um dieses Problem zu lösen, kann die JSD (siehe Definition 7) eingesetzt werden [Sha10]. Die Berechnung mit dieser Methodik ist in Abbildung 3.2 gezeigt.

Definition 7. *Gegeben seien zwei Wahrscheinlichkeitsverteilungen $p = (p_1, ..., p_N)$ und $q = (q_1, ..., q_N)$. Die JSD zwischen p und q wird wie folgt berechnet [Sha10]:*

$$JS(p,q) = \frac{1}{2}\left(KL\left(p||\frac{p+q}{2}\right) + KL\left(q||\frac{p+q}{2}\right)\right)$$

$$
\begin{aligned}
JS(T,T') &= \frac{1}{2}\left(KL\left(T||\frac{T+T'}{2}\right)+KL\left(T'||\frac{T+T'}{2}\right)\right)\\
JS(T,T') &= \frac{1}{2}\left(\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{1}{4}}{2}}\right)+\frac{3}{8}\cdot log_2\left(\frac{\frac{3}{8}}{\frac{\frac{1}{4}+\frac{3}{8}}{2}}\right)+\frac{3}{8}\cdot log_2\left(\frac{\frac{3}{8}}{\frac{\frac{1}{2}+\frac{3}{8}}{2}}\right)\right)+\\
&\quad\frac{1}{2}\left(\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{1}{4}}{2}}\right)+\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{3}{8}}{2}}\right)+\frac{1}{2}\cdot log_2\left(\frac{\frac{1}{2}}{\frac{\frac{1}{2}+\frac{3}{8}}{2}}\right)\right)\\
JS(T,T') &\approx 0,0155\\
JS(T,T'') &= \frac{1}{2}\left(KL\left(T||\frac{T+T''}{2}\right)+KL\left(T''||\frac{T+T''}{2}\right)\right)\\
JS(T,T'') &= \frac{1}{2}\left(\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{1}{4}}{2}}\right)+\frac{3}{8}\cdot log_2\left(\frac{\frac{3}{8}}{\frac{\frac{1}{2}+\frac{3}{8}}{2}}\right)+\frac{3}{8}\cdot log_2\left(\frac{\frac{3}{8}}{\frac{\frac{1}{4}+\frac{3}{8}}{2}}\right)\right)+\\
&\quad\frac{1}{2}\left(\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{1}{4}}{2}}\right)+\frac{1}{2}\cdot log_2\left(\frac{\frac{1}{2}}{\frac{\frac{1}{2}+\frac{3}{8}}{2}}\right)+\frac{1}{4}\cdot log_2\left(\frac{\frac{1}{4}}{\frac{\frac{1}{4}+\frac{3}{8}}{2}}\right)\right)\\
JS(T,T'') &\approx 0,0155
\end{aligned}
$$

Abbildung 3.2.: Berechnung des Wertes für t-Closeness (für Tabelle 3.7) mittels JSD

3.5. Slicing

Das Slicing stellt neben Generalisierung und Unterdrückung ein weiteres Verfahren zur Anonymisierung eines Datenbestandes dar. Das Ergebnis lässt sich allerdings nicht mit den bereits vorgestellten Maßen k-Anonymität, l-Diversität und t-Closeness messen. Anstatt einer Verallgemeinerung von Werten der Attribute wird bei diesem Konzept auf eine Permutation von Werten gesetzt, sodass die Daten dadurch anonymisiert werden. Allerdings enthalten diese noch konkrete Werte, die eine statistische Auswertung vereinfachen [LLZM12].

Beim Slicing wird eine Relation R in m vertikale und n horizontale Relationen aufgeteilt. Die entstandenen $n \cdot m$ Teilrelationen werden anschließend in sich permutiert und wieder zu einer anonymisierten Relation R' zusammengefügt [GH15]. Auf diese Weise ist es auch möglich, stark korrelierende Attribute zusammen zu halten und trotzdem ein gutes Maß an Anonymisierung zu gewährleisten.

In der nachfolgenden Tabelle 3.10 ist erkennbar, wie die Daten durch den Slicing-Algorithmus verändert werden. Dabei wurden innerhalb der Teilrelationen Permutationen durchgeführt und anschließend die Ergebnisse wieder zu einer Relation R' zusammengefügt. In der Teilrelation aus den Zeilen 1-5 und den Spalten *age* und *education* wurden Zeile 1 mit Zeile 4, Zeile 3 mit Zeile 2 und Zeile 5 im Ring getauscht. Die Vertauschungen werden dabei zufällig vorgenommen, allerdings müssen Tupel einzelner Teilrelationen in sich unverändert bleiben.

Zeile	age	education	maritalStatus	capitalGain	race	sex
1	39	Bachelors	Never-married	2174	White	Male
2	31	Master	Never-married	14084	White	Female
3	42	Bachelors	Married-civ-spouse	5178	White	Male
4	30	HS-grad	Married-civ-spouse	5013	White	Male
5	30	Bachelors	Married-civ-spouse	2407	White	Male
Zeile	age	education	maritalStatus	capitalGain	race	sex
6	44	HS-grad	Divorced	14344	White	Female
7	44	Bachelors	Married-civ-spouse	15024	White	Male
8	32	HS-grad	Married-civ-spouse	7688	White	Male
9	17	10th	Never-married	34095	White	Female
10	28	Some-college	Married-civ-spouse	4064	White	Female

Tabelle 3.9.: ResultSet der Anfrage:
SELECT age, education, maritalStatus, capitalGain, race, sex **from** adult **where** capitalGain>0 **limit** 10;.
Ein horizontaler Split wurde nach Zeile 5 durchgeführt und vertikale Splits jeweils nach *education*, *capitalGain* und *race*.

Zeile	age	education	maritalStatus	capitalGain	race	sex
1	30	HS-grad	Married-civ-spouse	5178	White	Male
2	42	Bachelors	Married-civ-spouse	5013	White	Male
3	30	Bachelors	Never-married	2174	White	Male
4	39	Bachelors	Married-civ-spouse	2407	White	Female
5	31	Master	Never-married	14084	White	Male
6	28	Some-college	Married-civ-spouse	15024	White	Male
7	17	10th	Divorced	14344	White	Female
8	44	Bachelors	Never-married	34095	White	Female
9	32	HS-grad	Married-civ-spouse	4064	White	Female
10	44	HS-grad	Married-civ-spouse	7688	White	Male

Tabelle 3.10.: Ergebnisrelation R' aus Tabelle 3.9 nach Anwendung des Slicing-Algorithmus.

Es sollte beachtet werden, dass nach der Veröffentlichung der Daten auch angegeben wird, in welcher Weise die Daten hier verändert wurden. Bei einer Veränderung der Daten für eine gestellte Datenbankanfrage muss die erfolgte Anonymisierung in den Metadaten festgehalten werden, damit auch bestimmt werden kann, welche Datenbankanfragen (Auswertungen) nach dem Slicing überhaupt noch sinnvoll durchgeführt werden können (anderen Anfragen / Auswertungen müssen vom System entsprechend verhindert werden). Diese müssen dann im Zuge des Provenance Managements weiter verarbeitet werden. Sollte dies nicht geschehen, kann es sehr schnell zu Missinterpretationen kommen, da die Daten scheinbar noch personenbeziehbar sind, beziehungsweise der Quasi-Identifikator noch ohne sichtliche Veränderung enthalten ist.

3.6. Differential Privacy

Differential Privacy ist wie k-Anonymität, l-Diversität oder auch t-Closeness ein Maß für den Grad der Anonymisierung von Daten. Grundsätzlich geht es darum, dass sichergestellt wird, dass das Hinzufügen oder Entfernen eines Datensatzes keine deutliche Änderung des Ergebnisses herbeiführt [Dwo08]. Dies kann durch eine Manipulation der Ausgabe oder je nach Anfrage auch durch eine ausreichend große Datenbasis erfüllt werden. Eine Manipulation kann beispielsweise durch das Hinzufügen von Rauschen erzeugt werden. Somit können die Unterschiede durch den neuen oder fehlenden Datensatz ausgeglichen werden. Es ist allerdings darauf zu achten, dass kein symmetrisches Rauschen bezüglich der Grundwerte eingesetzt wird, da dies durch eine gewisse Anzahl an Anfragen herausgefiltert werden könnte [Dwo08].

Definition 8. *Eine Zufallsfunktion K erfüllt ϵ-differential-Privacy, wenn für alle Datensätze D_1 und D_2, die sich in höchstens einem Tupel unterscheiden, gilt [Dwo09]:*

$$S \subseteq Wertebereich(K)$$
$$Pr[K(D_1) \in S] \leq e^{\epsilon} \cdot Pr[K(D_2) \in S]$$

D_1 und D_2 können dabei beliebig getauscht werden. Der Parameter ϵ gibt dabei den Informationsverlust an.

Eine Person wird durch eine Anonymisierung, welche Differential Privacy erfüllt, sehr gut geschützt, da die Anfrage keine Rückschlüsse auf den Wert des sensiblen Attributs der Person zulässt, obwohl der Datensatzes der Person in der Tabelle enthalten ist. Der Unterschied, ob der Datensatz enthalten ist oder nicht, wird nach Anwendung der Anonymisierung nicht mehr wahrnehmbar sein [Dwo09]. Der Einfluss des Rauschens kann über den Wert von ϵ reguliert werden. Je kleiner der Wert, desto höher die Anonymisierung. Typischerweise wird der Wert unter 1 liegen [Fri10].

4. Datenbank-Intrusion-Detection-Systeme

Unabhängig von den in Kapitel 3 vorgestellten Methoden zur Anonymisierung von Daten muss die Datenbank natürlich auch gegen Angriffe abgesichert sein. Hierfür eignet sich neben der sicheren Systemkonfiguration und einer Firewall beispielsweise ein Intrusion-Detection-System (IDS). Dieses dient dazu, eventuelle Angriffe oder Eindringversuche zu erkennen. Ein Intrusion-Prevention-System (IPS) geht einen Schritt weiter und versucht den Eindringversuch auch aktiv abzuwehren. In diesem Kapitel sollen aktuelle Möglichkeiten der Nutzung von Intrusion-Detection- und Intrusion-Prevention-Systemen im Bereich von Datenbankmanagementsystemen vorgestellt werden.

Die hier vorgestellten Systeme arbeiten nicht wie klassische Intrusion-Detection-Systeme auf Open System Interconnect Layer (OSI-Layer) 3 oder 4, sondern auf OSI-Layer 5. Dadurch sind den Systemen deutlich mehr datenbankspezifische Informationen zugänglich, die entsprechend ausgewertet werden können.

Die Hauptanforderungen für Datenbank-Intrusion-Detection-Systeme sind folgende [San14]:

- Erstellung und Definition von Nutzerprofilen welche das Standardverhalten der Nutzer repräsentieren. Für diese Anlernphase muss sichergestellt werden, dass keine bösartigen Anfragen im System auflaufen. Weiterhin müssen Angriffssignaturen zuverlässig erkannt werden. In der Architektur des PArADISE-Frameworks ließe sich dies gut in der Datenschutz-Policy einarbeiten. Darin werden auch Werte für den Grad der Anonymisierung übergeben. Durch eine Erweiterung können entsprechende Profile hinterlegt werden.
- Durch die gegebenen Profile und Signaturen muss definiert werden können, welche Modelle und Techniken die Geschwindigkeit und Performance des Systems maximieren. Dabei muss auch eine maximale Genauigkeit erreicht werden.
- Es muss möglich sein, den Systemstatus sicher zu überwachen und potentielle Angriffe entsprechend zu melden.
- Es sollte eine Möglichkeit gefunden werden, aufgedeckte Angriffe zu stoppen oder bereits präventiv noch vor der Ausführung zu verhindern.

Hat ein Angreifer Zugriff auf das System erlangt, kann er verschiedene Absichten verfolgen. Neben der dauerhaften Veränderung oder Verfälschung von Daten wäre es auch denkbar, dass der Angreifer es auf das Stehlen von Daten abgesehen hat. Als weitere Möglichkeit bietet sich dem Angreifer die Störung der Erreichbarkeit der Datenbank an. Dabei wird die Datenbank sehr stark mit aufwendigen Anfragen

konfrontiert, sodass das System zusammenbricht und auch auf die regulären Anfragen nicht mehr antworten kann [San14]. Dies wird als sogenannte Denial-of-Service-Attacke (DoS-Attacke) bezeichnet. Im Rahmen des PArADISE-Projektes wäre es bösartigen Angreifern zuzutrauen, dass der Heimanschluss einer Person, die am Programm der Sturzerkennung teilnimmt, per DoS-Attacke ausgelastet wird und so eine Kommunikation mit einem Cloud-Dienst nicht mehr möglich ist. Schwierige Anfragen zur Erkennung, ob ein Sturz vorliegt, wären so nicht mehr ausführbar. Dies könnte dazu führen, dass ein möglicher Sturz unerkannt bleibt. Für einen derartigen Angriff bräuchte der Angreifer natürlich vergleichsweise viel Vorwissen, allerdings wäre ein Angriff dieser Art für die entsprechende Person im Zweifel lebensbedrohlich.

Neben der regulären Sicherung des Datenbanksystems durch die Einschränkung der Rechte für einzelne Nutzer, kann der Einsatz eines IDS einen zusätzlichen Schutz bringen. Es ist beispielsweise nicht möglich, durch Zugriffsberechtigungen für einen Datenbanknutzer ein zeitliches Profil für die Zugriffe zu definieren. Hier könnte ein IDS wie in Absatz 4.1.1 beschrieben zum Einsatz kommen. Es ist damit feststellbar, ob die Zugriffe den zeitlichen Strukturen folgen und somit im Normbereich liegen. Abweichungen können effektiv detektiert und gemeldet werden. Die im Folgenden vorgestellten Intrusion-Detection-Systeme sind daher als Sicherheitserweiterung der Datenbankmanagementsysteme zu betrachten, da sie auch eng mit dem Datenbankmanagementsystem (DBMS) zusammenarbeiten.

Neben Angriffsszenarien und Schutzmaßnahmen kann auch der Angreifer kategorisiert werden. Ein Angriff kann durch einen autorisierten Nutzer erfolgen, welcher über gültige Zugangsdaten zur Datenbank verfügt. Als Angreifer handelt dieser Nutzer mit böswilliger Absicht. Es könnte allerdings auch einem externen Angreifer gelungen sein, die Zugangsdaten eines Nutzers zu stehlen und diese über ein offenes Backend zu nutzen. Es bleibt weiterhin die Möglichkeit, dass der Angreifer von extern ohne Zugangsdaten versucht, direkten Zugang zur Datenbank zu erlangen. Dies ist beispielsweise durch SQL-Injection-Angriffe denkbar. Je nach Szenario des Angreifers muss auch das Verteidigungssystem entsprechend ausgelegt sein [San14].

4.1. Klassifikation von Datenbank-Intrusion-Detection-Systemen

In diesem Abschnitt werden verschiedenste Techniken zur Erkennung von bösartigen Anfragen an die Datenbankmanagementsysteme vorgestellt. Diese erlauben eine Klassifizierung der IDSe. Im Abschnitt 4.1.8 werden die einzelnen Analysetechniken tabellarisch übersichtlich zusammengefasst.

4.1.1. Zeitbasierte Analyse

Bei dieser Technik wird insbesondere auf zeitliche Kriterien geachtet. Neben der regulären Zeitspanne zwischen Nutzeraktionen wird vor allem auch auf die Ausführungszeit von Anfragen geachtet. Sollte eine Anfrage ausgeführt werden, deren Ausführungszeit deutlich von der normalen abweicht, kann die Anfrage abgewiesen werden oder zumindest ein Alarm ausgelöst werden. Ebenso kann ein Alarm abgesetzt werden, wenn ein Objekt wiederholt mit einem Update verändert werden soll, obwohl dies in der aktuellen Periode bereits vorgenommen wurde. Besonders günstig ist diese Art von Überwachung bei Sensordaten. Diese werden regelmäßig aktualisiert und die Ausführungszeit ist dabei meist sehr ähnlich.

Abweichungen können daher leicht erkannt werden [LSS00].

Dank der zeitlichen Analyse kann diese Art von IDS auch als IPS eingesetzt werden. Schadhafte oder unerwünschte Anfragen können mittels der zeitlichen Analyse aufgespürt und vor der Ausführung blockiert werden [San14].

4.1.2. Abhängigkeits- und relationale Analyse

IDSe dieser Art beobachten neben den regulären Nutzeraktionen auch insbesondere die entsprechenden Daten, die für Anfragen genutzt werden. Neben den Kommandos werden so auch Attributeinschränkungen, Spalten und natürlich Tabellen in die Bewertung einbezogen. Um abschätzen zu können, ob eine Aktion zulässig ist, wird der Abstand des Wertepaares, aus Aktion und beeinflussten Daten zu denen der bereits gelernten Wertepaare, berechnet. Sollte die Distanz zu groß sein, so wird die Aktion als unzulässig angesehen. Die Analyse erfolgt dabei vor allem auf syntaktischer Ebene. Hierbei gibt es verschiedene Implementierungen, die sich hauptsächlich auf der syntaktischen Analyse der Anfragen unterscheiden [Kam08].

Systeme dieser Art können auch als vollwertiges IPS eingesetzt werden, da die Analyse für jede einzelne Anfrage ausgeführt wird und somit ein Blockieren der Anfrage, sobald sie nicht die entsprechenden Kriterien erfüllt, möglich ist [San14]. Dabei muss allerdings darauf geachtet werden, dass nicht mit Daten aus Log-Dateien gearbeitet wird, sondern alle Anfragen direkt geprüft werden oder solange blockiert werden, bis sie freigegeben sind.

4.1.3. Sequenzbasierte Analyse

Bei der sequenzbasierten Analyse wird vor allem die Reihenfolge der ausgeführten Anfragen betrachtet. In der Lernphase wird diese analysiert und in Ereignisse mit entsprechender Länge eingeteilt. Diese können unter Umständen noch verkürzt werden, um normales Nutzerverhalten darzustellen. Im laufenden Betrieb wird ausgewertet, wie Anfragen sich von den Anfragemustern der Lernphase unterscheiden, um auf dieser Basis eine Entscheidung treffen zu können, ob es sich um eine zulässige Aktion handelt. Je nach Ausrichtung der Datenbank ist es dabei möglich, unterschiedliche Parameter mit differenziertem Einfluss zu nutzen, sodass die Performance erhöht wird [Kun10].

Durch die Analyse der Anfragefolgen kann ein derartiges System bei der Feststellung einer unerwünschten Abfolge von Anfragen diese auch unterbinden. Allerdings schlägt die Erkennung unter Umständen erst nach der erfolgreichen Ausführung einiger Anfragen an. Damit ist lediglich ein partieller Einsatz als IPS möglich [San14].

4.1.4. Sequenzbasierte Analyse mit Integration von Abhängigkeiten

Diese Technik setzt neben der Analyse der Sequenzen von Anfragen vor allem auf Abhängigkeiten einzelner Anfragen. Eine SQL-Anfrage wird daher mit drei Mengen assoziiert. Die Lese-Menge umfasst Attribute, die vor dem Update gelesen wurden, die Pre-Write-Menge enthält Attribute, die vor dem eigentlichen Update geschrieben wurden und die Post-Write-Menge enthält Attribute, die nach dem Update im Zuge dessen geschrieben wurden. Um dieses Modell noch weiter zu vervollständigen, kann auch die Sensitivität eines Attributes angegeben werden [Sri06]. Mit Hilfe dieser Daten ist es ebenfalls möglich, eine Anfrage als zulässig oder unzulässig einzustufen.

In Bezug auf den Einsatz als IPS verhält sich die sequenzbasierte Analyse mit der Integration von Abhängigkeiten genau wie die sequenzbasierte Analyse, da auch hier Anfragefolgen betrachtet werden. Eine Erkennung ist daher unter Umständen erst nach einigen erfolgreich ausgeführten Anfragen möglich [San14].

4.1.5. Ergebnisorientierte Analyse

Die Analyse der Syntax ist eine Variante, um bösartige Anfragen zu erkennen. Bei der statistischen Analyse wird vor allem auf die zugegriffenen Daten geachtet und daraus das Benutzerprofil erstellt. Bei jeder Anfrage wird aus der Ergebnisrelation ein Vektor berechnet, der statistische Daten über die Anfrageergebnisse enthält. Diese Vektoren werden zu Clustern zusammengefasst und Nutzern zugeordnet. Sobald der Nutzer eine neue Anfrage stellt, wird geprüft, ob diese in das Cluster passt oder nicht und somit kann eine abweichende Handlung von den Routinen festgestellt werden. Diese Technik ist vor allem dazu geeignet, Angreifer von Innen zu stoppen, die Zugangsdaten zum System besitzen und versuchen, damit Daten aus der Datenbank illegal zu extrahieren [Mat10].

Aufgrund dessen sind IDSe dieser Art allerdings größtenteils nicht als IPS einsetzbar, da die Analyse auf der Veränderung der Daten in der Datenbank erfolgt. Auswertungen sind daher immer nur nach der Ausführung der Anfragen möglich. Lediglich eine Einschätzung aufgrund der angeforderten Spalten kann vor Ausführung eingesetzt werden [San14].

4.1.6. Informationstheoretische Analyse

Diese Methode setzt auf die Nutzung verschiedener Parameter. Für jedes Datenobjekt gibt es ein Tupel aus n Variablen, die beispielsweise *IP-Adresse*, *Nachrichtengröße*, oder ähnliches enthalten. Um diese Daten bewerten zu können, wird die Berechnung von Entropie eingesetzt. Dabei ist es möglich, auch die vorangegangenen Daten in die Berechnungen einfließen zu lassen. Es kann daher festgestellt werden, ob es sich um eine zulässige oder unzulässige Aktion handelt. Je öfter sich eine Aktion wiederholt, desto wahrscheinlicher wird sie zulässig sein [Lee01].

Ähnlich wie bei der statistischen Analyse eignet sich diese Art von IDSen nur sehr bedingt zum Einsatz als IPS. Sie kann aber ebenfalls mit Analyse der angeforderten Spalten teilweise zur Prävention eingesetzt werden [San14].

4.1.7. Anfragebasierte Analyse

Diese Art von IDSen eignet sich besonders gut für Anfragen, die in der Struktur vorbereitet sind und mit Werten aus Formularen gefüllt werden. Es wird ein Baum der SQL-Anfrage erzeugt. Dieser Parse-Baum wird dann vor und nach der Einsetzung der Formulardaten verglichen. Sollte die Struktur deutlich verändert sein, kann auf eine bösartige Anfrage geschlossen werden, da gutartige Anfragen, in der Regel, die Struktur nur unwesentlich verändern. Aufgrund dieser Muster ist es möglich, neue Anfragen einschätzen zu können. Dies kann auch noch mit Machine-Learning-Algorithmen verknüpft werden, um eine stetige Verbesserung zu erhalten. Insbesondere SQL-Injection-Angriffe lassen sich so sehr gut erkennen [Boc09].

Da die Analyse der Anfragen hier auf der Anfrage selbst basiert, können Systeme dieser Art ohne Einschränkung als IPS eingesetzt werden und bieten die Möglichkeit als bösartig erkannte Anfragen zu blockieren und die Ausführung zu verweigern [San14].

4.1.8. Zusammenfassung der Analysetechniken

		Analyseelemente			**Intrusion-Prevention-Möglichkeit**
Technik	**Verweis**	**Syntax**	**Spaltenzugriff**	**Ergebnis**	
Zeitbasierte Analyse	[LSS00]	✓			Ja
Abhängigkeits- und relationale Analyse	[Kam08]	✓	✓		Ja
Sequenzbasierte Analyse	[Kun10]	✓			Partiell
Sequenzbasierte Analyse mit Integration von Abhängigkeiten	[Sri06]	✓	✓		Partiell
Ereignisorientierte Analyse	[Mat10]	✓	✓	✓	Nein
Informationstheoretische Analyse	[Lee01]	✓			Partiell
Anfragebasierte Analyse	[Boc09]	✓	✓		Ja

Tabelle 4.1.: Übersicht über verschiedene Intrusion-Detection-Systeme (nach [San14])

In Tabelle 4.1 sind alle Analyseverfahren mit entsprechenden Referenzen noch einmal aufgeführt. Es zeigt sich, dass bei allen Verfahren die Syntax betrachtet wird, während der Spaltenzugriff schon nur noch bei einzelnen Techniken betrachtet wird und in eine Bewertung einfließt. Lediglich die ereignisorientierte Analyse bietet neben Syntax und Spaltenzugriff auch eine Beurteilung aufgrund der Ergebnisrelation an. Dies führt allerdings dazu, dass diese Technik nicht als IPS eingesetzt werden kann.

4.2. Anwendung von Datenbank-Intrusion-Detection-Systemen

Je nach verwendetem DBMS bieten sich unterschiedliche IDSe mehr oder weniger an. Für den folgenden Abschnitt sollen die DBMSe in transaktionale Datenbankmanagementsysteme und Data Warehouses eingeteilt werden.

Transaktionale Systeme sind darauf ausgelegt, ein möglichst stark normalisiertes Datenbankschema zu besitzen und die Redundanz von Daten zu vermeiden. Dabei gibt es in diesen Systemen zumeist eine sehr große Zahl an Nutzern, die allerdings in der Regel nur recht einfache Anfragen stellen, die schnell verarbeitet werden können. Besonders wichtig ist die hohe Verfügbarkeit dieser Datenbanksysteme.

Data Warehouses sind im Gegensatz dazu meist nur von wenigen Nutzern frequentiert, die aber sehr rechenintensive Anfragen stellen, da hiermit unter anderem statistische Auswertungen erstellt werden, um gewisse Geschäftsentscheidungen besser abwägen zu können. Die Daten werden nicht möglichst redundanzarm gespeichert, sondern es gibt eine Faktentabelle und mehrere Dimensionstabellen. Es wird dabei besonders Wert darauf gelegt, dass die Daten so gespeichert werden, dass die statistischen Anfragen möglichst einfach und schnell berechnet werden können.

Ein generelles Problem bei IDSen für Datenbanken stellt die Analyse der Syntax dar. Aufgrund des extrem umfangreichen Standards von SQL können Anfragen sehr unterschiedlich syntaktisch formuliert sein und dennoch semantisch das gleiche ausdrücken. Einem Angreifer ist es somit allerdings auch möglich, mit einer nur leicht anderen Formulierung ein semantisch deutlich anderes Ergebnis abzufragen und somit zu versuchen, das IDS zu umgehen. Es sei weiterhin erwähnt, dass es sich bei dem Problem des Query Containment und der Äquivalenz von konjunktiven SQL-Anfragen um ein unentscheidbares Problem handelt [San14]. Um die Komplexität einzugrenzen, kann die SQL-Syntax auf die Relationenalgebra einschränkt werden. Auf diese Art kann der Umfang sehr deutlich reduziert werden, allerdings noch ausreichend Funktionalität bereitstellen.

Je nach Datenbanksystem, ob transaktional oder als Data Warehouse, bieten sich daher unterschiedliche Techniken an. Für transaktionale Datenbanken, die zumeist mit sehr vielen Nutzern und einem Satz mehr oder weniger starrer Anfragen auskommen, kann mit Hilfe der Syntax-Analyse eine sehr effiziente Technik zur Überwachung der Datenbank genutzt werden. So zum Beispiel auch für Assistenzsysteme, die auch in der Cloud betrieben werden. Diese nutzen für spezielle Situations- oder Handlungserkennung im Wesentlichen immer die gleiche Anfrage – im Gegensatz zu Data Warehouses.

Bei Data Warehouses sind diese Techniken aufgrund flexibler Anfragen so nicht einsetzbar. Zudem muss ein System nach Möglichkeit auch als IPS einsetzbar sein, da so eine Extraktion von wertvollem Wissen verhindert werden kann. Daher bieten sich für Data Warehouses besonders die Abhängigkeits- und relationale Analyse an [San14]. Trotz dieser Tatsache ist es bei dieser Art von Datenbankmanagementsystemen besonders schwierig eine korrekte Einschätzung zu treffen, ob die Anfrage legitim ist oder nicht.

Die Kombination einer Analyse der betroffenen Daten mit einer ergebnisorientierten Analyse kann besonders empfohlen werden, da so vor und nach der Anfrage eine Einschätzung vom System generiert werden kann. Diese Untersuchungen können in Kombination meist eine recht gute Bewertung abgeben [San14].

5. De-Anonymisierungsverfahren

Während in Kapitel 3 verschiedene Maße der Anonymisierung von Daten und auch zugehörige Methoden vorgestellt wurden, beschreibt dieses Kapitel, an welchen Stellen die entsprechenden Verfahren versagen. Zusätzlich werden Möglichkeiten beschrieben, die es für einen „Angreifer“ oder eine Person mit böswilliger Absicht gibt, um aus den anonymisierten Daten Rückschlüsse auf personenbezogene Daten zu gewinnen.

Grundsätzlich lassen sich zwei Ansätze unterscheiden. Bei anfragebasierten Verfahren geht es um eine Manipulation der Anfrage, sodass trotz einer Einschränkung der erlaubten Anfragen Informationen extrahiert werden können, die eigentlich nicht zugänglich sein sollten. Der datenbasierte Ansatz beschreibt hingegen Angriffe auf Daten, die durch eine Anfrage gewonnen wurden. In der Regel werden datenbasierte Verfahren genutzt, um Personen in großen Datenbeständen nicht identifizierbar zu machen.

In unserem Szenario sollen Handlungen von (meist bekannten) Personen in einer Wohnung nicht identifizierbar gemacht werden. Der „klassische“ Fall wäre es, Personen in einer Menge nicht identifizierbar zu machen. Weitere Informationen zu den gemachten Anpassungen in dieser Arbeit sind in Abschnitt 2.3 zu finden. Handlungen werden in Assistenzsystem durch Machine-Learning-Verfahren oder ähnliches erkannt. Solche Verfahren werden in PArADISE in SQL-Anfragen umgewandelt. Es ist also wichtig, erlaubte Anfragen zuzulassen (bestimmte Handlungen sollen erkannt werden) und andere Anfragen (die schützenswerte Handlungen erkennen) auszuschließen. Erlaubte Anfragen können beispielsweise in Sichten modelliert werden. Näheres dazu im folgenden Abschnitt.

5.1. Anfragebasierte De-Anonymisierungsverfahren

In diesem Abschnitt wird untersucht, ob und wie es möglich ist, Anfragen an das PArADISE-System lediglich mit Sichten zu beantworten. Um die Anfragen umzuschreiben, sodass lediglich Sichten genutzt werden, wird die als „Answering Queries using Views (AQuV)“ bekannte Technik genutzt. Es soll sichergestellt werden, dass nur die zulässigen Informationen den Bereich, in dem die Daten anfallen, verlassen. Es wird weiterhin geprüft, ob es trotz der vorausgewählten Sichten möglich ist, Informationen zu extrahieren, die nicht freigegeben werden sollten.

5.1.1. Sichtkonzept in Datenbanken

Sichten stellen virtuelle Relationen im Datenbanksystem dar. Durch die Nutzung lassen sich beispielsweise komplexe Anfragen vereinfachen, da Teilausdrücke der Anfrage durch eine Sicht erfüllt werden können. Zusätzlich sind Sichten für die Zugriffskontrolle sehr hilfreich, da der Zugriff auf eine Datenbank mit Sichten (engl. Views) sehr gut eingeschränkt werden kann.

Updates auf Sichten stellen allerdings eine besondere Herausforderung dar, oder sind gar unmöglich, da eine Update-Operation auf einer Sicht zum Beispiel dazu führen könnte, dass einzelne Attribute nicht ausgefüllt wurden, welche allerdings in den ursprünglichen Relationen benötigt werden. In Bezug auf das PArADISE-Projekt wäre dies allerdings zu vernachlässigen, da keine Updates auf den Sichten ausgeführt werden müssen.

Um Sichten bereits vor ankommenden Anfragen berechnet zu haben, können sogenannte materialisierte Sichten genutzt werden. Bei diesen muss allerdings auch beachtet werden, was passiert, wenn sich die ursprünglichen Relationen ändern. Es gibt dazu verschiedene Ansätze. Zum einen kann die Sicht bei jedem Update einer Relation, von welcher Daten aktualisiert werden, neu berechnet werden. Dies führt allerdings zu versteckten Kosten bei Update-Operationen auf der Datenbank. Zum anderen ist es möglich die materialisierte Sicht zu bestimmten Zeitpunkten zu aktualisieren. Dies führt wiederum zu dem Problem, dass es Zeitpunkte gibt, zu denen die Informationen der Sicht nicht aktuell sind.

Materialisierte Sichten sind allerdings bei Stromdaten, wie sie in Assistenzsystemen in Form von Sensordaten vorkommen, eher untypisch. Daher ist dieses Problem für das PArADISE-Projekt ebenfalls zu vernachlässigen.

5.1.2. Answering Queries using Views

Sowohl Sichten als auch Anfragen werden derzeit in der AQuV-Technik nur mit einfachen relationenalgebraischen Anfragen gebildet (Projektion, Verbund, einfache Selektionen). PArADISE benötigt aber in beiden Fällen auch komplexe SQL-Anfragen. Für diese scheitert die Technik aber.

Grundsätzlich wird die „Answering-Queries-using-Views“-Technik vor allem dafür eingesetzt, um Anfragen zu beschleunigen. Es ist möglich mit Hilfe von Sichten bereits aufwendige Joins vom Datenbanksystem im Hintergrund abarbeiten zu lassen, sodass das entsprechende Ergebnis direkt bei der Anfrage bereit steht. Dadurch können auch komplexere Anfragen schneller beantwortet werden. Es ist ebenfalls möglich, den Zugriff auf die Datenbank durch die Nutzung von Sichten gezielt einzuschränken. Es kann beispielsweise Nutzern gestattet werden, nur Anfragen an Sichten zu stellen und damit kann der Zugriff auf einzelne Attribute, die zum Beispiel als verbindendes Attribut zweier Tabellen für einen Join nötig sind, verweigert werden. Trotz dessen kann die Ergebnisrelation des Joins der beiden Tabellen benutzt werden.

Eine Anfrage nutzt lediglich Sichten, wenn hinter dem **FROM**-Teil der Anfrage nur diese referenziert werden. Die Beantwortung von Anfragen mit Hilfe von Sichten ist ein allgemeineres Problem, als die Frage, ob eine Anfrage einer anderen gleicht, da auch andere Sichten zur Beantwortung einer Anfrage zu Rate gezogen werden können und somit mehrere Sichten für eine Anfrage genutzt werden können. Um eine Anfrage in eine umzuwandeln, welche nur Sichten nutzt, gibt es verschiedene Algorithmen.

Die Gleichheit oder das Enthalten sein einer Anfrage in einer anderen wird durch folgende Definition beschrieben.

Definition 9. *Seien Q_1 und Q_2 Anfragen mit der gleichen Zahl an Attributen. Q_1 ist in Q_2 enthalten, notiert als $Q_1 \sqsubseteq Q_2$, wenn für jede Datenbank D gilt, $Q_1(D) \subseteq Q_2(D)$, wobei $Q(D)$ das Ergebnis der Anfrage darstellt. Q_1 ist äquivalent zu Q_2, notiert als $Q_1 \equiv Q_2$, wenn $Q_1 \sqsubseteq Q_2$ und $Q_2 \sqsubseteq Q_1$ gilt [DHI12].*

Zu beachten ist bei Definition 9, dass sich die Gleichheit auf die Anfragen und nicht auf das Ergebnis bezieht. Zwei nach Definition 9 äquivalente Anfragen werden für jede Datenbank D das gleiche Ergebnis liefern. Das Enthalten sein unterschiedlicher Anfragen ineinander ist bei negierten Attributen, Gruppierungen und Aggregaten in der Anfrage noch deutlich komplizierter zu prüfen [DHI12] und aktuell mit der AQuV-Technik nur mit Einschränkungen möglich. Im Rahmen dieser Arbeit wird dies aber aufgrund der Komplexität vernachlässigt und nicht näher betrachtet, da dies den Umfang übersteigen würde.

Die bedeutendste Frage beim Umschreiben einer regulären Anfrage in eine Anfrage, die lediglich Sichten verwendet ist, ob die Anfrage 1:1 das gleiche Ergebnis liefert oder ob lediglich ein Ergebnis erreicht wird, das möglichst nah an dem der ursprünglichen Anfrage liegt. Dies wird durch die folgenden beiden Definitionen beschrieben.

Definition 10. *Sei Q eine Anfrage und $V = \{V_1, ..., V_m\}$ eine Menge von definierten Sichten. Die Anfrage Q' ist äquivalent zu Q unter der Benutzung von V, wenn Q' nur die Sichten aus V nutzt und Q' äquivalent zu Q ist [DHI12].*

Definition 10 beschreibt, wann eine Anfrage als äquivalent zu einer anderen gilt.

Definition 11. *Sei Q eine Anfrage, $V = \{V_1, ..., V_m\}$ eine Menge von definierten Sichten und L eine Anfragesprache. Die Anfrage Q' enthält ein Maximum des Ergebnisses der Anfrage Q unter der Benutzung von V und Berücksichtigung von L, wenn Q' eine Anfrage in L ist und sich nur auf Sichten aus V bezieht, Q' in Q enthalten ist und es keine Umschreibung $Q_1 \in L$ gibt, sodass gilt $Q' \sqsubset Q_1 \sqsubseteq Q$ [DHI12].*

Definition 11 wiederum beschreibt, wann eine Anfrage eine andere maximal enthält. Dies bedeutet, dass die Ergebnisrelation der umgeschriebenen Anfrage so viele Tupel wie möglich der Ergebnisrelation der ursprünglichen Anfrage enthält.

Um eine vordefinierte Sicht für eine Anfrage nutzen zu können, müssen folgende Voraussetzungen erfüllt sein [Hal01].

- Es muss eine Abbildung ψ geben, die von den Tabellen, welche im **FROM**-Teil der Sicht von V genutzt werden, auf die Tabellen des **FROM**-Teils der Anfrage Q abbildet. Dabei muss jeder Tabellenname auf sich selbst abgebildet werden. Beim Einsatz von Multimengen-Semantiken muss ψ eine 1:1 Abbildung sein, während bei Semantiken mit einer Menge ohne Duplikate eine n:1 Abbildung möglich ist.
- V muss entweder die Join- und Selektionsattribute der Anfrage Q auf die Attribute der Tabelle in der Definitionsmenge von ψ nutzen, oder muss logisch schwächere Bedingungen an die Selektion stellen.
- V darf keine Attribute von der Projektion aus der Definitionsmenge von ψ ausschließen, die für die Selektion von Q nötig sind, außer diese Attribute lassen sich aus anderen Sichten, oder den originalen Tabellen (insofern diese zugänglich sind), wiederherstellen.

Es muss weiterhin beachtet werden, dass die Verwendung von Multimengen weitere Bedingungen nötig macht. Die Multiplizität von einzelnen Tupeln muss erhalten werden. Diese könnte zum Beispiel durch den Einsatz von **DISTINCT** verringert werden, oder durch zusätzliche Joins erhöht werden [Hal01]. Für die AQuV-Theorie stellt dies ein weiteres Problem dar, da diese sich zunächst nur auf Mengen bezieht, SQL allerdings Multimengen bietet.

5.1.3. Der MiniCon-Algorithmus

Der MiniCon-Algorithmus stellt eine Verbesserung zum Bucket-Algorithmus (siehe dazu [LRO96]) dar. Beide Algorithmen geben zu einer Anfrage und einer Menge von Sichten eine Menge an Anfrageplänen zurück, wobei diese dann die Umschreibung mit den Sichten enthalten. Ziel der Algorithmen ist die Beschleunigung der Umschreibung einer Anfrage, sodass nach Möglichkeit nur Sichten genutzt werden, dazu werden Unnötige ausgeschlossen. Sollte eine äquivalente Umschreibung unmöglich sein, so wird ein Anfrageplan ausgegeben, der so viele Elemente wie möglich der originalen Ergebnisrelation enthalten wird.

Grundsätzlich geht es sowohl beim Bucket-, als auch dem MiniCon-Algorithmus darum, den Suchraum zur Umschreibung der Anfragen einzuschränken. Der Bucket-Algorithmus zerlegt zu Beginn die Anfrage in Teilziele und sucht entsprechend passende Sichten für diese. Diese werden dann in einem zweiten Schritt kombiniert, um so die gesamte Anfrage erfüllen zu können. Es muss allerdings in einem letzten Schritt noch geprüft werden, ob die kombinierte Anfrage zu der ursprünglichen Anfrage passt. Problematisch beim Bucket-Algorithmus ist die Betrachtung der Teilziele in Isolation. Dadurch kommen zu viele der Sichten als potentiell Passende in die Buckets und der zweite Schritt, die Kombination dieser Sichten und das Prüfen, ob sie mit der originalen Anfrage übereinstimmen, wird sehr teuer, da wieder viele Kombinationen getestet werden müssen [PH01].

Der MiniCon-Algorithmus beginnt ähnlich wie der Bucket-Algorithmus, indem nach Zwischenzielen gesucht wird, die in Sichten und der originalen Anfrage enthalten sind. Sobald eine Abbildung von einem Zwischenziel der Anfrage und einer Sicht gefunden wurde, wird die Sichtweise des Algorithmus umgekehrt und es wird auf die Variablen in der Anfrage geachtet. Der Algorithmus zieht die Join-Prädikate

in die Betrachtung mit ein und sucht nach der minimalen zusätzlichen Menge an Zwischenzielen, die auf das Zwischenziel der Sicht abgebildet werden müssen. Das Tupel aus Zwischenzielen und Abbildungsinformationen wird als „MiniCon Description (MCD)“ bezeichnet. In der zweiten Phase werden die entsprechenden MCDs zu einer vollständigen Anfrage kombiniert, um die ursprüngliche Anfrage komplett zu erfüllen. Dabei ist zu beachten, dass aufgrund der aufwendigeren Vorarbeit die kombinierten Anfragen nicht mehr auf das Enthaltensein oder auf Gleichheit mit der originalen Anfrage geprüft werden müssen [PH01]. Dies führt zu einer deutlichen besseren Performance, vor allem bei einer höheren Anzahl an Sichten oder auch komplexeren Anfragen [PH01].

Definition 12. *Eine MCD C für eine Anfrage Q über eine Sicht V ist ein Tupel der Form $(h_C, V(Y)_C, \varphi_C, G_C)$, wobei folgendes gilt [PH01]:*

- *h_C ist der Kopf-Homomorphismus von V,*
- *$V(Y)_C$ ist das Ergebnis der Anwendung von h_C auf V (zum Beispiel $Y = h_C(A)$, wobei A die Kopf-Variablen von V sind),*
- *φ_C ist eine partielle Abbildung von Vars(Q) nach $h_C(Vars(V))$,*
- *G_C ist eine Teilmenge der Teilziele von Q, welche durch einige Teilziele in $h_C(V)$ und die Abbildung φ_C) abgedeckt sind.*

C kann nur dann für eine nicht redundante Umschreibung von Q genutzt werden, wenn die folgenden zwei Bedingungen erfüllt sind:

1. *Für jede Kopf-Variable (head variable) x aus Q, welche im Wertebereich von φ_C liegt, ist $\varphi_C(x)$ eine Kopf-Variable in $h_C(V)$.*
2. *Wenn $\varphi_C(x)$ eine benötigte Variable in $h_C(V)$ ist, dann gilt für jedes g, Zwischenziel von Q, das x enthält, dass alle Variablen in g auch im Wertebereich von φ_C liegen und $\varphi_C(g) \in h_C(V)$ gilt.*

Ein Kopf-Homomorphismus h auf einer Sicht V ist eine Abbildung h von $Vars(V)$ nach $Vars(V)$, welche eine Identitätsabbildung auf den benötigten Variablen ist und ungleiche Variablen ausgleichen kann [PH01]. Wie in Definition 12 zu sehen, ist in jeder MiniCon Description ein solcher Kopf-Homomorphismus enthalten.

Das folgende Beispiel nach [PH01] soll die Funktion des MiniCon-Algorithmus etwas verdeutlichen. Folgende Anfrage und Sichten sind vorhanden:

- Anfrage: $Q(x) :- behandelt(x, y), behandelt(y, x), gleiche\_Krankheit(x, y)$
- Sicht 1: $V_1(a) :- behandelt(a, b), behandelt(b, a)$
- Sicht 2: $V_2(c, d) :- gleiche\_Krankheit(c, d)$
- Sicht 3: $V_3(f, h) :- behandelt(f, g), behandelt(g, h), gleiche\_Krankheit(f, g)$

Mit der Anfrage soll herausgefunden werden, welche Ärzte die gleiche Krankheit haben und sich gegenseitig behandeln. In Tabelle 5.1 sind die fertigen MiniCon Descriptions aufgelistet. Sicht V_1 kann aufgrund von Bedingung 2 aus Definition 12 nicht genutzt werden, da nicht beide Variablen (a und b) im Kopf der Sicht auftauchen. Dies ist allerdings nötig, um die gewonnenen Tupel mit der Ergebnisrelation aus Teilziel 3 der Anfrage Q ($gleiche\_Krankheit(x, y)$) zu joinen und so ein Ergebnis für Q zu erhalten. Die MCD für V_2 benötigt im Kopf ($V(Y)$) beide Variablen (c und d), da mit dieser Sicht nur das Teilziel (G) 3 von der Anfrage Q erfüllt werden kann und beide Variablen für die Erfüllung der Teilziele 1 und 2 nötig sind. Sicht V_3 kann alle drei Teilziele von Q erfüllen und benötigt daher im Kopf nur eine Variable, wie Anfrage Q. Der Kopf-Homomorphismus ist hier nicht nur die identische Abbildung, sondern bildet beide Variablen (f und h) der Sicht auf eine Variable, f, ab.

$V(Y)$	h	φ	G
$V2(c, d)$	$c \to c, d \to d$	$x \to c, y \to d$	3
$V3(f, f)$	$f \to f, h \to f$	$x \to f, y \to f$	1,2,3

Tabelle 5.1.: Generierte MCDs zu dem Beispiel aus Absatz 5.1.3

In der zweiten Phase des Algorithmus werden die einzelnen MCDs zusammengesetzt. Um eine nicht redundante Umschreibung zu erreichen, muss folgende Bedingung erfüllt sein:

Definition 13. *Gegeben eine Anfrage Q, eine Menge an Sichten V und eine Menge an MCDs C für Q über den Sichten in V sind die einzigen nicht redundanten Kombinationen der MCDs von Q in der Form $C_1, ..., C_l$. Dabei müssen folgende zwei Bedingungen gelten:*

1. $G_{C_1} \cup ... \cup G_{C_l} = Subgoals(Q)$ *und*
2. *für jedes* $i \neq j$ *gilt* $G_{C_i} \cap G_{C_j} = \emptyset$.

Im obigen Beispiel wird der Algorithmus zuerst die MCD mit Sicht V_5 betrachten. Daher wird er nach MCDs suchen, welche Teilzeile 1 und/oder 2 abdecken, sodass die Bedingungen für das Zusammenfügen von MCDs, siehe Definition 13, erfüllt sind. Da es allerdings keine MCDs gibt, die dies erfüllen, wird die MCD aus Sicht V_5 verworfen. Als Nächste wird die MCD zu Sicht V_6 betrachtet. Diese erfüllt alle Teilziele und da es die letzte verfügbare MCD ist, ist es auch gleichzeitig die einzig mögliche Umschreibung zu Q. Die Ausgabe wäre daher $Q(x) :- V3(x, x)$.

5.1.4. Angriffsmöglichkeiten durch Answering Queries using Views

Im konkreten Beispiel könnte die „Answering-Queries-using-Views“-Technik für das PArADISE-Projekt eingesetzt werden. Dabei würden erlaubte Anfragen als Sichten definiert werden und der Zugriff würde auf Sichten begrenzt werden. Dies würde dazu führen, dass keine unerlaubten Attribute abgefragt werden könnten. Natürlich muss auch für die Veröffentlichung der Daten aus den Sichten eine Anonymisierung, zum Beispiel mittels Generalisierung, durchgeführt werden, um die Angriffstechniken aus Abschnitt 5.2 zu vereiteln. Weiterhin ist dies nötig, um die vorgegebenen Anonymisierungsmaße erfüllen zu können.

Es muss jede einzelne Sicht geprüft werden, ob mit dieser persönliche oder personenbeziehbare Daten veröffentlicht werden, um sie im Zweifel auszuschließen. Dazu muss auch das Gesamtbild aller zugänglichen Sichten betrachtet werden, da auch hier Joins über die Ergebnisse dieser Sichten denkbar wären, was wiederum zur Veröffentlichung unerwünschter Informationen führen könnte.

Aus Sicht des Angreifers wäre es nötig alle verfügbaren Sichten und auch die entsprechende Struktur der Datenbank zu kennen, um mit diesem Wissen Anfragen zu konstruieren, die mit den Sichten beantwortet werden können und zusätzlich Informationen freigeben, die so nicht sichtbar sein sollten.

5.2. Datenbasierte De-Anonymisierungsverfahren

Der datenbasierte Ansatz beschäftigt sich mit der Auswertung der Informationen, die auf Anfragen an das Datenbanksystem erhalten wurden. Die Auswertung erfolgt dabei unabhängig von der Anfrage.

5.2.1. Homogenitätsattacke

Dieser Angriff ist ohne aktives Eingreifen des Angreifers ausführbar. Er beschreibt ein großes Problem der k-Anonymität. Wie einleitend in Abschnitt 3.3 beschrieben, ist das Problem die geringe Vielfältigkeit in Bezug auf das sensitive Attribut [MKGV07]. Als Beispiel betrachten wir Tabelle 3.3. Der rot hervorgehobene q^*-Block der 2-anonymen Tabelle (Zeile 2 und Zeile 6) besitzt als Werte für das sensitive Attribut „capitalGain“ 14084 und 14344.

Für einen Nachbarn, einer der beiden Personen, ist es wichtig zu wissen, ob der Wert über 10000 liegt oder nicht. Er kann aufgrund der anderen Attribute sicherstellen, dass die Person in diesem roten q^*-Block liegt. Aufgrund der niedrigen Vielfältigkeit im sensitiven Attribut dieser Gruppe kann er schlussfolgern, dass der Wert für die Nachbarin entweder 14084 oder 14344 betragen muss und somit in beiden Fällen deutlich über 10000 liegt.

Das Maß der l-Diversität nimmt sich dieses Problems partiell an. Hier ist es wichtig, dass l verschiedene Werte im Wertebereich des sensitiven Attributs auftreten. Bei obigem Beispiel wäre dies allerdings auch schon wieder schwierig. Eine Tabelle mit Krankheiten soll als weiteres Fallbeispiel dienen. Es ist möglich, darin zu erkennen, dass eine Person in einem gewissen q^*-Block liegt, allerdings haben alle Personen in diesem Block unterschiedliche Arten von Krebs. Für den „Angreifer“ ist es lediglich interessant, ob die Person Krebs hat oder nicht. Damit wäre die entscheidende Information bereits offengelegt.

Um diesen Angriff deutlich schwieriger zu gestalten, muss t-Closeness mit einem ausreichend kleinen t gewährleistet sein. Das t gibt dabei an, wie stark die Verteilung des sensitiven Attributes im q^*-Block von der Verteilung über alle Werte des sensitiven Attributes abweichen darf. Für l-Diversität sollte zusätzlich ein entsprechend großes l gewählt werden. Jeder q^*-Block muss dann mindestens l unterschiedliche Werte enthalten.

5.2.2. Hintergrundwissen

Starkes Hintergrundwissen kann ein großes Problem sein und schnell zu einer Offenlegung von persönlichen Daten führen. K-Anonymität ist gegen dieses Problem anfällig, aber auch l-Diversität kann bei starkem Hintergrundwissen keinen ausreichenden Grad an Anonymisierung bieten [MKGV07].

Als Beispiel soll Tabelle 3.5 dienen. Tom weiß, dass seine Kommilitonin im Krankenhaus liegt und hat die Tabelle mit den anonymisierten Daten gefunden. Er weiß, dass Ihre Postleitzahl 18069 ist und sie 1990 geboren ist. Aufgrund dieser Informationen kann er darauf schließen, dass ihre Daten im grünen q^*-Block erfasst sind und sie damit entweder ein gebrochenes Bein hat oder an Inkontinenz leidet. Da er sie zwei Tage später in der Vorlesung sieht und sie keinen Gips am Bein trägt, allerdings etwas häufiger auf Toilette muss, schlussfolgert er, dass sie an Inkontinenz leidet.

Dieses Problem lässt sich vermindern, indem die Werte für k-Anonymität und l-Diversität entsprechend groß gewählt werden. Ein Angreifer braucht so beispielsweise immer l-1 Informationen, um aus den vorliegenden Daten Rückschlüsse auf ein einzelnes Tupel ziehen zu können, beziehungsweise die anderen ausschließen zu können. Es muss beachtet werden, dass Werte semantisch ähnlich sein können. So kann ein gebrochener Arm und ein gebrochenes Bein nur durch das Sehen der Person ausgeschlossen werden, was vielleicht nur als ein Wert für das Hintergrundwissen gezählt wird, aber doppelt gewertet werden muss.

5.2.3. Angriff durch Sortierung der Tupel

Dieser Angriff ist besonders bei der Veröffentlichung unterschiedlicher Versionen einer Tabelle denkbar. Das Problem stellt die Sortierung der Tupel dar. In einer relationalen Datenbank sind die Tupel eigentlich nach dem Relationenmodell eine Menge und damit ungeordnet. Allerdings sind viele Dateiorganisationsformen und Zugriffspfade (zum Beispiel ein B-Baum) sortiert und die Ergebnismenge damit nach einem bestimmten Attribut geordnet [Swe02]. Diese Ordnung wird häufig auch durch die Anwendungen der verschiedenen Mechanismen zur Anonymisierung nicht verändert. Dadurch ist es unter Umständen möglich, zwei unterschiedliche Veröffentlichungen einer Tabelle, die in sich anonym sind, zu verknüpfen und so auf private Daten Rückschlüsse ziehen zu können. Dies soll durch das Beispiel in Tabelle 5.2 verdeutlicht werden.
Ganz links ist die Tabelle zu sehen, wie sie im Datenbank-System hinterlegt sein könnte. Der Quasi-Identifikator soll in diesem Beispiel durch die Kombination von *Geburtsjahr* und *Postleitzahl* gebildet werden. Auf ein sensitives Attribut wurde zur Vereinfachung verzichtet. Sollten die beiden Tabellen (Mitte und Rechts) veröffentlicht werden, ist es möglich, ohne Probleme eine Verknüpfung beider herzustellen und damit auch den Quasi-Identifikator zu rekonstruieren. Eine Offenlegung von sensiblen Daten ist damit möglich.

Diese Angriffsmöglichkeit kann allerdings besonders einfach vereitelt werden, indem die einzelnen Tupel der Tabellen vor der Veröffentlichung zufällig sortiert werden. Gilt k-Anonymität für k=2, ist ein eindeu-

Geburtsjahr	Postleitzahl
1994	18055
1983	18057
1965	18055
1963	18055
1975	18059
1977	18057
1955	18181

Geburtsjahr	Postleitzahl
1980-1994	18055
1980-1994	18057
1965-1979	18055
1950-1964	18055
1965-1979	18059
1965-1979	18057
1950-1964	18181

Geburtsjahr	Postleitzahl
1994	18000-18199
1983	18000-18199
1965	18000-18199
1963	18000-18199
1975	18000-18199
1977	18000-18199
1955	18000-18199

Tabelle 5.2.: Ursprungstabelle (links) und jeweils eine der Spalten anonymisiert, sodass k-Anonymität für k=2 erfüllt ist. Quasi-Identifikator ist *Geburtsjahr* und *Postleitzahl*.

tiges Verbinden der Tupel nicht mehr möglich.

Dieses Vorgehen ist auch besonders wichtig, wenn der direkte Zugriff auf die Datenbank ermöglicht wird. Sollte ein Angreifer einen Datenbankzugang haben und somit eigene Anfragen stellen können, gibt es ein Problem, wenn die Anonymisierung, wie im Beispiel des Treibers, der im Rahmen des PArADISE-Projektes entwickelt wird, umgesetzt wird. Hier werden die Ergebnisse auf enthaltene Quasi-Identifikatoren geprüft und anonymisiert, sollte einer enthalten sein. Wenn kein QI enthalten ist, wird das Ergebnis ohne Änderung an den Anfragenden zurückgegeben. Werden die Tupel nicht zufällig sortiert, könnte der Angreifer, vorausgesetzt er sucht nur nach Werten, die in einer Tabelle stehen, einfach die Attribute einzeln abfragen, und aus den Ergebnissen eine Tabelle zusammensetzen. Dies würde ebenfalls zur ungewollten Veröffentlichung von persönlichen Informationen führen.

5.2.4. Angriff durch komplementäre Veröffentlichungen

Typischerweise stellt der Quasi-Identifikator eine echte Teilmenge der Attribute der veröffentlichten Tabelle dar. Angenommen es wird parallel dazu noch eine andere Variante der Tabelle veröffentlicht. Sollten in der zweiten Veröffentlichung nicht alle Attribute der ersten Veröffentlichung im Quasi-Identifikator enthalten sein, so kann ein Angreifer die Tabellen eventuell über Attribute verknüpfen, welche nicht im Quasi-Identifikator enthalten sind. Dadurch kann es auch zur Offenlegung persönlicher Daten, also zur Kompromittierung des Quasi-Identifikators, kommen [Swe02]. Um dies zu umgehen, müssen entweder alle Attribute der ursprünglichen Tabelle in den Quasi-Identifikator der darauffolgenden Tabelle aufgenommen werden, oder die Tabelle basiert auf der ersten Veröffentlichung. Es geht also darum, die veröffentlichten Daten mit den „restlichen“ Informationen in der Datenbank zu verknüpfen, oder auch externe Datenbanken für eine Verknüpfung von Informationen zu nutzen. Dies könnte zur Identifikation von einzelnen Personen führen.

5.2.5. Zeitbasierter Angriff

In der Regel entwickeln sich Datenbestände fortwährend weiter und es kommen neue Datensätze hinzu und alte werden gelöscht. Angenommen eine Tabelle T wird zum Zeitpunkt 0 und unter Erfüllung von k-Anonymität als Tabelle KT_0 veröffentlicht. Zu einem späteren Zeitpunkt t, Tupel wurden in der Zwischenzeit eingefügt und/oder gelöscht, wird Tabelle T wieder unter Erfüllung von k-Anonymität als Tabelle KT_t veröffentlicht. Dabei kann es passieren, dass für KT_t ein anderer Quasi-Identifikator gewählt wird. Eine Verbindung beider Tabellen kann zur Offenlegung von privaten Daten führen. Um dies zu vermeiden, muss die gesamte Menge der Attribute von KT_0 in den Quasi-Identifikator von KT_T aufgenommen werden, oder die darauffolgenden Veröffentlichungen basieren direkt auf Tabelle KT_0 [Swe02].

6. Durchführung eines Angriffs

Eine allgemein nutzbare Implementierung für Angriffe gegen Datenbankveröffentlichungen, nach den in Kapitel 5 vorgestellten Verfahren, ist nicht oder nur extrem schwer zu erreichen. Das größte Problem dabei ist die fehlende Flexibilität eines implementierten Algorithmus. Es müssen sehr viele Faktoren beachtet werden und sollte der Angriff komplett durch die Maschine erfolgen, müssten viele schwer zu parametrisierende Werte vom Angreifer entsprechend dem Algorithmus übergeben werden. Dies würde den Umfang der hier vorliegenden Arbeit überschreiten.

6.1. Bewertung des ResultSets

Um einen möglichen Angriff trotzdem leichter zu gestalten, beziehungsweise schneller die Möglichkeiten oder das Potential eines Angriffs zu erkennen, wurde ein Verfahren entwickelt, das Analysen auf einem vorliegenden ResultSet durchführt. Damit wird einem Angreifer die Entscheidung für die Wahl möglicher Angriffsvektoren erleichtert. Es ist somit schneller möglich, einen erfolgreichen Angriff durchzuführen und entsprechende Informationen zu gewinnen, beziehungsweise abzuschätzen, wie aufwendig ein Angriff auf die vorliegenden Daten ist.

6.1.1. QI-Erkennung mittels TopDownBottomUp-Mechanismus

Quasi-Identifikatoren wurden bereits in Definition 1 beschrieben und abgegrenzt. An dieser Stelle soll zusätzlich der Begriff des minimalen Quasi-Identifikators genauer bestimmt werden.

Definition 14. *Q_T sei ein Quasi-Identifikator aus einer endlichen Menge von Attributen $Q_T := \{A_i, ..., A_j\}$ einer Tabelle T mit einer endlichen Menge von Attributen $A := \{A_1, A_2, ..., A_n\}$. Hierbei gilt $Q_T \subseteq A$. Q_T ist ein minimaler Quasi-Identifikator, wenn es keine Menge $B \subset Q_T$ gibt, wobei B ebenfalls ein Quasi-Identifikator ist [GH14].*

Der TopDownBottomUp-Mechanismus setzt sich aus dem TopDown-Mechanismus und dem BottomUp-Mechanismus zur Erkennung von Quasi-Identifikatoren zusammen. Beide Mechanismen haben entsprechende Vor- und Nachteile, was sich vor allem in der Laufzeit zeigt. Der BottomUp-Mechanismus bietet zusätzlich die Garantie, dass gefundene QIs bereits minimal sind [GH14].

Um einen Quasi-Identifikator zu erkennen, muss geprüft werden, wie viele einzigartige Tupel der Attributkombination, aus den Attributen des potentiellen QIs, es gibt. Diese Anzahl wird mit der Gesamtanzahl an Tupeln in der Relation ins Verhältnis gesetzt. Wird dabei der vorgegebene Schwellwert nicht unterschritten, gilt die Attributkombination als QI. Weiterhin gilt, sobald eine identifizierende Attributmenge

in einer größeren Obermenge enthalten ist, so ist auch diese Obermenge eine identifizierende Attributmenge. Dies gilt ebenfalls für Quasi-Identifikatoren [GH14]. Sobald daher ein QI aus weniger Attributen gefunden wird, der Teilmenge eines QIs aus dem TopDown-Ansatz ist, werden alle Quasi-Identifikatoren gestrichen, die mehr Elemente beinhalten. Damit kann auch beim Einsatz des TopDown-Ansatzes Minimalität gewährleistet werden.

Im TopDownBottomUp-Algorithmus werden die einzelnen Verfahren kombiniert. Dabei können die jeweiligen Vorteile genutzt werden. Während der BottomUp-Ansatz besonders gut für Quasi-Identifikatoren ist, die aus möglichst wenigen Attributen bestehen, kann das TopDown-Verfahren schon im ersten Schritt feststellen, ob es überhaupt einen Quasi-Identifikator oberhalb des gewählten Schwellwerts gibt. Zur Kombination gibt es eine Wichtungsfunktion, die bestimmt, welcher Algorithmus den nächsten Schritt ausführt. Die jeweiligen Ergebnisse der Teilschritte werden zwischen den Algorithmen ausgetauscht und es wird somit sichergestellt, dass möglichst wenige Attributkombinationen getestet werden müssen [GH14]. Für einen Angreifer ist es besonders wichtig, die minimalen Quasi-Identifikatoren zu finden, da er so nur einen minimalen Aufwand betreiben muss, um mit den Daten selbst oder zusätzlich gesuchten Informationen auf entsprechende persönliche Daten Rückschlüsse zu ziehen.

6.1.2. Analyse des ResultSets

Das Programm zur Analyse des ResultSets arbeitet lediglich auf den zurück gelieferten Werten und stellt für die Bewertung keine weiteren Anfragen an die Datenbank. Dies ist wichtig, um in Kapitel 4 vorgestellte Techniken zu umgehen, beziehungsweise die Erkennung von Angriffen zu erschweren, sollten Intrusion-Detection-Systeme genutzt werden. Das ResultSet wird daher komplett in einer eigenen Datenstruktur abgespeichert. Auf diesen Daten werden anschließende Analysen ausgeführt. Um dies konsequent umzusetzen, wurde der TopDownBottomUp-Ansatz modifiziert, da dieser die Auswertungen beispielsweise mit Hilfe der Datenbank über entsprechende Anfragen ausführt. Die entsprechenden Funktionen wurden um- beziehungsweise neu geschrieben.

Nachfolgend sind die drei wichtigsten Funktionen des Programms abgebildet. An diesen lässt sich der Ablauf zu Großteil erklären.

```
1  readDatabase()
2      Werte fuer Verbindung zu Datenbank einlesen
3      Verbindung von Treiber-Manager anfordern
4      Statement erstellen und Anfrage ausfuehren
5      Ueber Spalten iterieren
6          Spalten in eigener Datenstruktur initialisieren
7      Zeiger zurueck vor ersten Eintrag des Ergebnisses setzen
8      Ueber Ergebnisrelation iterieren
9          Ergebnisse in eigene Datenstruktur schreiben
10     Metadaten zu eigener Datenstruktur hinzufuegen
11     Datenbankverbindung schliessen
```

Der Funktion `readDataBase` werden ein Objekt vom Typ `Result`, ein Name für eine Datei mit den Einstellungen für die Datenbankverbindung und die Anfrage an sich übergeben. Um die Übersichtlichkeit zu erhöhen, ist diese Funktion als Pseudocode dargestellt. Im `Result`-Objekt wird das Ergebnis der Anfrage lokal zwischengespeichert. Dieses wird später auch an die Funktionen zur Auswertung weiter gegeben. Nach dem Einlesen der Werte für die Verbindung mit der Datenbank wird die Verbindung aufgebaut. Mit Hilfe dieser ist es möglich, die Anfrage auszuführen. Aus der Ergebnisrelation werden zuerst alle Spalten ausgelesen und diese im `Result`-Objekt angelegt. Danach werden Schritt für Schritt die einzelnen Werte eingelesen und in das Objekt geschrieben. Hierfür dient die `pushValue`-Funktion. Zum Schluss werden die gewonnenen ResultSet-Metadaten ebenfalls zum `Result`-Objekt hinzugefügt und die Verbindung wird geschlossen.

```
writeResultSet()
    Ausgabe der Ergebnisrelation wenn Flag gesetzt
    Aufruf doAnalysis()
```

In der Funktion `writeResultSet` wird geprüft, ob das Ergebnis auf der Konsole ausgegeben werden soll. Anschließend wird hier die Funktion zur Analyse des Ergebnisses aufgerufen, die hier `doAnalysis` heißt. Sie wird direkt auf dem `Result`-Objekt ausgeführt und benötigt daher keine Parameter, da die nötigen Werte im Objekt selbst stehen.

```
public void doAnalysis() throws SQLException,Exception{
  Double threshold = 95.0;
  TopDownBottomUpDetector tdbud = new TopDownBottomUpDetector(this, threshold);
  Map<Set<String>,Double> qis = new HashMap<Set<String>,Double>();

  try {
    qis = tdbud.topDownBottomUpDetection();
    while((qis.size()<=0) && (threshold >= 10.0)){
      threshold -= 5.0;
      System.out.println("Keine QIs gefunden, setze Threshold auf " + threshold +
    " %");
      tdbud.setThreshold(threshold);
      qis = tdbud.topDownBottomUpDetection();
    }
  } catch (Exception e) {
    e.printStackTrace();
  }

  try {
    int offset = 0;
    for(int rsmdsSize=0; rsmdsSize<rsmds.size(); rsmdsSize++){
      for(int j=1; j<=rsmds.get(offset).getColumnCount(); j++){
        System.out.println("Spalte " + rsmds.get(offset).getColumnLabel(j) + ":");
        String type = "";
        try {
          type = rsmds.get(offset).getColumnTypeName(j);
        } catch (SQLException e) {
```

```
27            throw e;
28          }
29
30          if (type.equals("INT")){
31            int rangeMin = 0;
32            int rangeMax = 0;
33            int rangeInt = 0;
34            boolean correctValue = false;
35            System.out.print("\tMoechten Sie einen Wertebereich angeben? (1 = Ja | 0
      = Nein): ");
36            while(!correctValue){
37              if (scanner.hasNextInt()) {
38                rangeInt = scanner.nextInt();
39                if ((rangeInt >= 0) && (rangeInt <= 1)){
40                  if (rangeInt == 1) {
41                    boolean correctValue2 = false;
42                    boolean correctValue3 = false;
43                    System.out.print("\tBitte geben Sie das Minimum ein: ");
44                    while(!correctValue2){
45                      if (scanner.hasNextInt()) {
46                        rangeMin = scanner.nextInt();
47                        correctValue2 = true;
48                        } else {
49                          System.out.print("\tFalscher Werte, Eingabe bitte
      wiederholen: ");
50                          scanner.next();
51                        }
52                    }
53                    System.out.print("\tBitte geben Sie das Maximum ein: ");
54                    while(!correctValue3){
55                      if (scanner.hasNextInt()) {
56                        rangeMax = scanner.nextInt();
57                        correctValue3 = true;
58                        } else {
59                          System.out.print("\tFalscher Werte, Eingabe bitte
      wiederholen: ");
60                          scanner.next();
61                        }
62                    }
63                    correctValue = true;
64                  } else {
65                    correctValue = true;
66                  }
67                }
68                } else {
69                  System.out.print("\tFalscher Werte, Eingabe bitte wiederholen: ");
70                  scanner.next();
71                }
72            }
73            /* Ausgabe der berechneten Werte auf der Konsole */
74          }
```

```
75            /* Ausfuehrungen fuer DATE und STRING gekuerzt, aber aehnlich */
76          }
77          offset ++;
78        }
79      } catch (SQLException e) {
80        e.printStackTrace();
81      }
82      /*QIs ausgeben*/
83  }
```

Der Hauptanteil der Arbeit des Programms wird in der Funktion `doAnalysis` durchgeführt. Zuallererst wird mit der Bestimmung der QIs begonnen. Sollte mit einem Threshold von 95 Prozent kein QI gefunden werden, so wird der Wert Schritt für Schritt um jeweils fünf Prozentpunkte gesenkt, bis eine entsprechende Attributkombination gefunden wurde, oder der Wert auf das Minimum von fünf Prozent sinkt. Ab fünf Prozent ist die Attributkombination ohnehin nicht mehr von Belang.

Im nächsten Schritt wird über alle ResultSets iteriert. Es könnten durch mehrere Anfragen auch mehrere Ergebnisse vorliegen. In einer zweiten Schleife werden alle Spalten einzeln aufgerufen und es wird versucht, mit Hilfe der Metadaten zum ResultSet den Typ der Spalte zu bestimmen. Der Typ ist wichtig, um entsprechend der gespeicherten Werte eine sinnvolle Auswertung vornehmen zu können. Beispielhaft ist oben die Auswertung für Spalten mit dem Typ `INT` gezeigt. Es wird zuerst nach einem Wertebereich gefragt, bevor der Durchschnitt inklusive Minimum und Maximum bestimmt werden. Sollte eine Eingrenzung des Wertebereichs vorgenommen worden sein, wird auch angezeigt, wie viele Werte ausgeschlossen wurden. Für die Bestimmung der Anzahl unterschiedlicher Werte in einer Spalte wird bei einer Begrenzung des Wertebereichs der Wert mit, als auch ohne Beschränkung angegeben.

Die Bestimmung von Durchschnittswerten findet auch bei Spalten vom Typ `DATE` statt. Für Werte des Typs `STRING` wird darauf verzichtet, da dies keine verwertbaren Ergebnisse liefern würde. Eine Erweiterung um weitere Datentypen wäre vergleichsweise einfach möglich.

Die Analyseergebnisse geben die Möglichkeit, einen eventuellen Angriffsvektor einfacher auswählen zu können beziehungsweise eine fundierte Entscheidung zu treffen. Dafür werden je nach Spalte die entsprechenden Werte analysiert. Dabei wird die Anzahl unterschiedlicher Werte und auch die durchschnittliche Häufigkeit je Wert angegeben. Bei numerischen Werten kann der Nutzer zusätzlich den Durchschnitt, das Minimum und das Maximum einsehen (siehe Abbildung 6.1). Damit ist der Angreifer in der Lage einzusehen, wie stark die Werte in einer Spalte differenziert sind und wo Techniken aus Kapitel 5.2 am günstigsten angewendet werden können.

Weiterhin wurde die Möglichkeit geschaffen den Wertebereich einzugrenzen. Dies ermöglicht es, bestimmte fehlerhafte Werte von den Berechnungen auszuschließen. Beispielsweise sei eine Spalte mit Daten gegeben, wie lang ein Mitarbeiter in einer Firma angestellt ist. Bei unbefristeten Verträgen wird ein Wert von *9999-01-01* eingetragen. Dieser soll selbstverständlich nicht im Durchschnitt auftauchen, da der ansonsten sehr stark verfälscht wird und nicht mehr genutzt werden kann. Nach dem Ausschluss

```
Folgende Beispiele stehen zur Auswahl:
(1) SELECT * FROM dept_emp LIMIT 10;
(2) Select * from dept_emp NATURAL JOIN employees NATURAL JOIN departments LIMIT 1000;
(3) Select * from adult LIMIT 1000;
(4) Select age,education,workclass,maritalStatus,relationship,race,sex,nativeCountry,whatever from adult LIMIT 1000;
(5) SELECT emp_no,dept_no FROM dept_emp LIMIT 10; kombinitert mit SELECT from_date,to_date FROM dept_emp LIMIT 10;
(6) Eigene Anfrage (Name für Settings-Datei muss auch angegeben werden)

Bitte geben Sie eine Zahl für das entsprechende Beispiel ein: 1
Wollen Sie die Ergebnisrealtion auf der Konsole ausgeben? (1 = Ja | 0 = Nein): 1
Lese resources/employeedb-local (...)
Server DSN: //localhost/employees?
Driver Class: com.mysql.jdbc.Driver
Sun Jan 15 14:58:52 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According t
Anfrage: SELECT * FROM dept_emp LIMIT 10;
emp_no   |      dept_no  |      from_date        |      to_date  |

10001    |      d005     |      1986-06-26       |      9999-01-01       |
10002    |      d007     |      1996-08-03       |      9999-01-01       |
10003    |      d004     |      1995-12-03       |      9999-01-01       |
10004    |      d004     |      1986-12-01       |      9999-01-01       |
10005    |      d003     |      1989-09-12       |      9999-01-01       |
10006    |      d005     |      1990-08-05       |      9999-01-01       |
10007    |      d008     |      1989-02-10       |      9999-01-01       |
10008    |      d005     |      1998-03-11       |      2000-07-31       |
10009    |      d006     |      1985-02-18       |      9999-01-01       |
10010    |      d004     |      1996-11-24       |      2000-06-26       |
Spalte emp_no:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 10005,50 (min:10001.0 | max: 10010.0 | Anzahl ausgeschlossener Werte: 0.0)
        Anzahl unterschiedlicher Werte: 10.0 (Durchschnittliche Häufigkeit je Wert: 1,00
Spalte dept_no:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 6.0 (Durchschnittliche Häufigkeit je Wert: 1,67
Spalte from_date:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 1991-07-21 (min:1985-02-18 | max: 1998-03-11 | Anzahl ausgeschlossener Werte: 0)
        Anzahl unterschiedlicher Werte: 10.0 (Durchschnittliche Häufigkeit je Wert: 1,00
Spalte to_date:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 1
        Bitte geben Sie das frühest mögliche Datum ein (JJJJ-MM-TT): 1900-01-01
        Bitte geben Sie das spätest mögliche Datum ein (JJJJ-MM-TT): 2020-01-01
        Durchschnitt: 2000-07-13 (min:2000-06-26 | max: 2000-07-31 | Anzahl ausgeschlossener Werte: 8)
        Anzahl unterschiedlicher Werte im angegebenen Bereich: 2.0 (Durchschnittliche Häufigkeit je Wert im angegebenen Bereich: 1,00
        Anzahl unterschiedlicher Werte: 3.0 (Durchschnittliche Häufigkeit je Wert: 3,33
------------------------------------------------------------
Gefundene QIs:

QI: [from_date]                 mit 100.0 Prozent eindeutig.
QI: [emp_no]                    mit 100.0 Prozent eindeutig.
```

Abbildung 6.1.: Ausgabe einer Beispielanfrage an eine Datenbank um einen Angriffsvektor zu wählen.

erhält der Nutzer eine Angabe über den Durchschnitt, das Minimum und das Maximum im angegebenen Wertebereich. Außerdem wird die Anzahl der unterschiedlichen Werte mit der durchschnittlichen Häufigkeit mit und ohne Einschränkung des Wertebereichs angegeben. Es ist somit auch leicht ablesbar, wie viele Werte ausgeschlossen wurden. Dies kann sehr gut in Abbildung 6.1 gesehen werden. Hier wurde der Wertebereich der Spalte *to_ date* entsprechend eingeschränkt.

Um eine besonders aussagekräftige Spalte oder auch Kombination von Spalten zu finden, mit der Tupel möglichst einfach identifiziert werden können, wird zusätzlich der TopDownBottomUp-Mechanismus genutzt. Dieser kann zur Erkennung von Quasi-Identifikatoren eingesetzt werden. In diesem Fall wird das Ergebnis auf das Vorhandensein von eben solchen untersucht. Gefundene Quasi-Identifikatoren werden mit der dazugehörigen Wahrscheinlichkeit ausgegeben, mit der sie eindeutig sind.

Für den TopDownBottomUp-Mechanismus ist ein Schwellwert nötig. Dieser gibt an, ab wie viel Prozent eine Attributkombination als Quasi-Identifikator gilt. Je höher der Prozentsatz des gefundenen QIs, desto besser ist die Kombination seiner Attribute für einen Angriff geeignet. Wird der Schwellwert allerdings zu hoch gewählt, ist es möglich, dass keine Quasi-Identifikatoren mehr gefunden werden. Wird er zu tief angesetzt, werden schnell alle Spalten einzeln als QI gefunden und es werden entsprechend keine komplexeren, aber besser identifizierende QIs, mehr gefunden. Um trotz dessen ein möglichst gutes Ergebnis

```
Folgende Beispiele stehen zur Auswahl:
(1) SELECT * FROM dept_emp LIMIT 10;
(2) Select * from dept_emp NATURAL JOIN employees NATURAL JOIN departments LIMIT 1000;
(3) Select * from adult LIMIT 1000;
(4) Select age,education,workclass,maritalStatus,relationship,race,sex,nativeCountry,whatever from adult LIMIT 1000;
(5) SELECT emp_no,dept_no FROM dept_emp LIMIT 10; kombinitert mit SELECT from_date,to_date FROM dept_emp LIMIT 10;
(6) Eigene Anfrage (Name für Settings-Datei muss auch angegeben werden)

Bitte geben Sie eine Zahl für das entsprechende Beispiel ein: 4
Wollen Sie die Ergebnisrealtion auf der Konsole ausgeben? (1 = Ja | 0 = Nein): 0
Lese resources/adultdb-local (...)
Server DSN: //localhost/adult?
Driver Class: com.mysql.jdbc.Driver
Sun Jan 15 15:02:16 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL
Anfrage: Select age,education,workclass,maritalStatus,relationship,race,sex,nativeCountry,whatever from adult LIMIT 1000;
Keine QIs gefunden, setze Threshold auf 90.0 %
Keine QIs gefunden, setze Threshold auf 85.0 %
Spalte age:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 38,05 (min:17.0 | max: 90.0 | Anzahl ausgeschlossener Werte: 0.0)
        Anzahl unterschiedlicher Werte: 66.0 (Durchschnittliche Häufigkeit je Wert: 15,15
Spalte education:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 16.0 (Durchschnittliche Häufigkeit je Wert: 62,50
Spalte workclass:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 7.0 (Durchschnittliche Häufigkeit je Wert: 142,86
Spalte maritalStatus:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 7.0 (Durchschnittliche Häufigkeit je Wert: 142,86
Spalte relationship:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 6.0 (Durchschnittliche Häufigkeit je Wert: 166,67
Spalte race:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 5.0 (Durchschnittliche Häufigkeit je Wert: 200,00
Spalte sex:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 2.0 (Durchschnittliche Häufigkeit je Wert: 500,00
Spalte nativeCountry:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 29.0 (Durchschnittliche Häufigkeit je Wert: 34,48
Spalte whatever:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Anzahl unterschiedlicher Werte: 2.0 (Durchschnittliche Häufigkeit je Wert: 500,00
-------------------------------------------------------
Gefundene QIs:

QI: [nativeCountry, education, workclass, race, relationship, age, maritalStatus, whatever]                    mit 85.0 Prozent eindeutig.
QI: [nativeCountry, education, workclass, race, sex, age, maritalStatus, whatever]                  mit 85.0 Prozent eindeutig.
QI: [workclass, education, race, sex, relationship, whatever, age]                mit 85.0 Prozent eindeutig.
QI: [nativeCountry, education, workclass, race, sex, relationship, age, maritalStatus]              mit 86.0 Prozent eindeutig.
QI: [nativeCountry, education, workclass, sex, relationship, age, maritalStatus, whatever]                    mit 85.0 Prozent eindeutig.
```

Abbildung 6.2.: Anfrage an eine Datenbank, wobei erst bei einem Threshold von 85-90% ein QI gefunden wurde.

zu erhalten, wurde die Funktion so implementiert, dass mit einem Schwellwert von 95 Prozent begonnen wird und dieser Wert, sollten keine Ergebnisse gefunden werden, Schritt für Schritt jeweils um fünf Prozent abgesenkt wird, bis ein entsprechendes Ergebnis gefunden werden kann. Somit wird sichergestellt, dass immer ein möglichst genauer QI gefunden wird. In Abbildung 6.2 ist zu sehen, wie der Schwellwert zweimal abgesenkt wird, da vorher keine QIs gefunden wurden.

6.1.3. Bewertung der Analyseergebnisse

Die Analyseergebnisse geben besondere Hilfestellung bei einem Angriff über komplementäre Veröffentlichungen. Sobald ein Quasi-Identifikator mit ausreichend hoher Wahrscheinlichkeit gefunden wurde, kann nach weiteren Veröffentlichungen mit den entsprechenden Attributen des QIs gesucht werden. Wenn entsprechende Ergebnisse gefunden werden, können die Tupel verknüpft und so peu à peu neue Informationen gewonnen werden, um den Datenbestand immer mehr zu erweitern. Durch die Garantie des TopDownBottomUp-Mechanismus, einen minimalen Quasi-Identifikator zu finden und das abgestufte Herabsetzen des Schwellwerts für einen solchen, wird die Wahrscheinlichkeit des Findens eindeutiger Tupel erhöht und zugleich der Suchaufwand verringert.

Es wird auch eine große Hilfe im Bereich der Angriffe mit Hintergrundwissen gegeben. Durch die Angabe, wie häufig Werte in den einzelnen Spalten, sprich Attributen, vorkommen, kann die Auswahl nach dem zu suchenden Hintergrundwissen eingeschränkt werden. Der Angreifer wählt hier gezielt eine Spalte, deren Werte möglichst selten vorkommen und braucht so nur ein Minimum an Hintergrundinformationen zu sammeln, um entsprechende Rückschlüsse auf die persönlichen Informationen ziehen zu können. Durch die Möglichkeit, den Wertebereich weiter einzuschränken, ist es weiterhin möglich, noch schneller gezieltes Hintergrundwissen zu suchen.

Bezüglich des zeitbasierten Angriffs kann auch eine Hilfestellung gegeben werden. Beide Veröffentlichungen müssen einzeln analysiert werden. Der Angreifer sucht nach einem gefundenen Quasi-Identifikator, der möglichst genau ist und in beiden Tabellen vorkommt. Alternativ kann nach einer Spalte gesucht werden, in der das durchschnittliche Auftreten eines Wertes möglichst klein ist. Das garantiert, dass beim Verbinden beider Tabellen möglichst wenige Kombinationen innerhalb des Attributs, beziehungsweise der Attributkombination, möglich sind, auf dem die Tabellen verbunden werden.

```
Folgende Beispiele stehen zur Auswahl:
(1) SELECT * FROM dept_emp LIMIT 10;
(2) Select * from dept_emp NATURAL JOIN employees NATURAL JOIN departments LIMIT 1000;
(3) Select * from adult LIMIT 1000;
(4) Select age,education,workclass,maritalStatus,relationship,race,sex,nativeCountry,whatever from adult LIMIT 1000;
(5) SELECT emp_no,dept_no FROM dept_emp LIMIT 10; kombinitert mit SELECT from_date,to_date FROM dept_emp LIMIT 10;
(6) Eigene Anfrage (Name für Settings-Datei muss auch angegeben werden)

Bitte geben Sie eine Zahl für das entsprechende Beispiel ein: 5
Wollen Sie die Ergebnisrealtion auf der Konsole ausgeben? (1 = Ja | 0 = Nein): 1
Lese resources/employeedb-local (...)
Server DSN: //localhost/employees?
Driver Class: com.mysql.jdbc.Driver
Sun Jan 15 15:00:27 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According t
Anfrage: SELECT emp_no,dept_no FROM dept_emp LIMIT 10;
Lese resources/employeedb-local (...)
Server DSN: //localhost/employees?
Driver Class: com.mysql.jdbc.Driver
Sun Jan 15 15:00:27 CET 2017 WARN: Establishing SSL connection without server's identity verification is not recommended. According t
Anfrage: SELECT from_date,to_date FROM dept_emp LIMIT 10;
emp_no   |      dept_no  |      from_date        |      to_date  |

10001    |      d005     |      1986-06-26       |      9999-01-01       |
10002    |      d007     |      1996-08-03       |      9999-01-01       |
10003    |      d004     |      1995-12-03       |      9999-01-01       |
10004    |      d004     |      1986-12-01       |      9999-01-01       |
10005    |      d003     |      1989-09-12       |      9999-01-01       |
10006    |      d005     |      1990-08-05       |      9999-01-01       |
10007    |      d008     |      1989-02-10       |      9999-01-01       |
10008    |      d005     |      1998-03-11       |      2000-07-31       |
10009    |      d006     |      1985-02-18       |      9999-01-01       |
10010    |      d004     |      1996-11-24       |      2000-06-26       |
Spalte emp_no:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 10005,50 (min:10001.0 | max: 10010.0 | Anzahl ausgeschlossener Werte: 0.0)
        Anzahl unterschiedlicher Werte: 10.0 (Durchschnittliche Häufigkeit je Wert: 1,00
Spalte dept_no:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 1
        Folgende Werte sind in der Spalte vorhanden: d005 | d007 | d004 | d003 | d008 | d006
        Geben Sie an, welche Werte ausgeschlossen werden sollen (Leerzeichen zur Trennung): d003 d006
        Anzahl unterschiedlicher Werte im angegebenen Bereich: 4.0 (Durchschnittliche Häufigkeit je Wert im angegebenen Bereich: 2,00
        Anzahl unterschiedlicher Werte: 6.0 (Durchschnittliche Häufigkeit je Wert: 1,67
Spalte from_date:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 1991-07-21 (min:1985-02-18 | max: 1998-03-11 | Anzahl ausgeschlossener Werte: 0)
        Anzahl unterschiedlicher Werte: 10.0 (Durchschnittliche Häufigkeit je Wert: 1,00
Spalte to_date:
        Möchten Sie einen Wertebereich angeben? (1 = Ja | 0 = Nein): 0
        Durchschnitt: 8399-04-22 (min:2000-06-26 | max: 9999-01-01 | Anzahl ausgeschlossener Werte: 0)
        Anzahl unterschiedlicher Werte: 3.0 (Durchschnittliche Häufigkeit je Wert: 3,33
------------------------------------------------------------
Gefundene QIs:

QI: [from_date]                 mit 100.0 Prozent eindeutig.
QI: [emp_no]                    mit 100.0 Prozent eindeutig.
```

Abbildung 6.3.: Ausgabe bei einer Kombination von zwei unterschiedlichen Anfragen.

Für Angriffe über die Sortierung der Tupel wurde die Möglichkeit geschaffen, zwei unterschiedliche ResultSets miteinander zu verbinden, ohne ein entsprechendes Attribut für einen Join zu besitzen. Die Verknüpfung wird direkt im Programm übernommen und wurde so implementiert, dass beide Tabellen direkt nebeneinander geschrieben werden und somit zu einer verknüpft werden. Die Nutzung eines Intrusion-Detection- oder auch Intrusion-Prevention-Systems auf Serverseite kann den Angriff sehr stark erschweren oder vereiteln, da für diese Art von Angriff mehrere Anfragen nötig sind. Unabhängig davon kann ein IDS-IPS-System natürlich auch Angriffe mit nur einer Anfrage verhindern, wenn die entsprechenden Kriterien für den Nutzer, in Kombination mit der Anfrage, nicht erfüllt sind. Grundsätzlich geht es für den Angreifer bei dieser Technik darum, die Anfrage in einzelne kleine und scheinbar unschädliche Anfragen aufzuspalten und dann die entsprechenden Ergebnisse wieder zusammenzusetzen. Dies funktioniert allerdings nur, wenn alle Tupel aus einer Relation stammen. Ansonsten wäre ein Zusammensetzen der Tupel nicht möglich, da die Sortierung nicht mehr korrekt wäre. Um zu prüfen, ob die Sortierung der Tupel verändert wurde, kann einfach eine Anfrage mehrfach ausgeführt werden und die Ergebnisse verglichen werden. Beispielhaft ist dies in Abbildung 6.3 gezeigt, wobei die Anfrage aus Abbildung 6.1 in zwei Anfragen umgeschrieben wurde und die Ergebnisse anschließend wieder kombiniert wurden.

Bei Angriffen auf dem Weg der Homogenitätsattacke kann das Programm derzeit keine Hilfestellung geben. Näheres dazu im nachfolgenden Absatz.

6.2. Weitere Angriffsmöglichkeiten

Neben den Möglichkeiten, die Ergebnisrelation einer Anfrage zu analysieren und daraus möglichst viele Informationen zu extrahieren, oder die Ergebnisse zweier Anfragen direkt zu verknüpfen, gibt es noch die Variante, über die Homogenität von Attributen Informationen zu extrahieren.

Um einen Angreifer auf Seite der Homogenitätsattacke zu unterstützen, könnte eine Wissensdatenbank aufgebaut werden, um entsprechende Attributwerte einer Kategorie zuordnen zu können. Für die einzelnen Kategorien könnten anschließend Generalisierungsbäume hinterlegt werden. Mit diesen Informationen kann die Generalisierung von Seite des Angreifers eingesetzt werden. Es wäre mit einer derartigen Datenbank möglich, Attribute solange zu generalisieren, bis in der gefundenen Tabelle eindeutige Tupel zu finden sind. Dafür müssten doppelte Tupel entsprechend nach jedem Generalisierungsschritt gestrichen werden. Anschließend erhält der Angreifer ein entsprechend allgemeineres Ergebnis, was allerdings noch immer ausreichen könnte, um persönliche Informationen aufzudecken.

Dabei ist es denkbar, dass der Angreifer einem entsprechenden Hilfsprogramm das ResultSet, das entsprechend interessante Tupel und auch das sensitive Attribut übergibt. Für das Tupel müssten alle bekannten Attributwerte angegeben werden, wobei dies dann nicht auf nur ein Tupel zurückzuführen ist. Die Generalisierung würde dann so lange durchgeführt werden, bis die entsprechende Gruppe von Tupeln einen eindeutigen Wert beim sensitiven Attribut annimmt.

7. Zusammenfassung und Ausblick

Ziel der Arbeit war die Kategorisierung und Anwendbarkeit von De-Anonymisierungs-Techniken. Dabei sollten die Möglichkeiten diskutiert werden, anonymisierte Daten wiederherstellen zu können. Der Aufwand der signifikanten Techniken wurde in dieser Arbeit ebenfalls betrachtet, wie auch die Voraussetzungen, welche dafür erfüllt sein müssen. Es wurde an beispielhaften SQL-Anfragen gezeigt, dass Angriffe automatisiert unterstützt werden können. Eine vollständige Automation des Angriffsprozesses wäre ein sehr hoher Programmieraufwand, da die Techniken sehr vielfältig sind und es für den Angreifer nötig ist, individuell auf den Datenbestand reagieren zu können. Zusätzlich wurde die „Answering-Queries-Using-Views"-Technik betrachtet, welche eingesetzt werden kann, um den Datenbankzugriff sehr stark zu beschränken.

Um die entsprechenden De-Anonymisierungs-Techniken verstehen zu können, ist es notwendig, auch die verwendeten Konzepte der Anonymisierung zu betrachten. Dabei wurden in Kapitel 3 Techniken vorgestellt, die vorgenommene Anonymisierung quantifizierbar und somit messbar machen zu können. Es wurde auch gezeigt, dass es bessere Strategien gibt, die Daten zu schützen, als einfach den Zugriff auf die Datenbank zu beschränken. Das PArADISE-Framework bietet hier sehr gute Voraussetzungen und erlaubt es, die einzelnen Teilnehmer an der Datenverarbeitung in unterschiedliche Schichten einzuteilen und so feingranular Zugriffsrechte zu vergeben. Durch den Anfrageprozessor ist es weiterhin möglich, die Anfragen zu modifizieren und Ergebnisse, entsprechend der Richtlinien, zu anonymisieren. Um die reguläre Nutzung eines Zugangs der Datenbank zu überwachen, bietet sich weiterhin der Einsatz eines IDSs an. Sollen bestimmte Attribute, die für Joins nötig sind, nicht von außen zugänglich sein, so kann die „Answering-Queries-Using-Views"-Technik genutzt werden. Der Zugang wird dabei auf vordefinierte Sichten eingeschränkt. Diese Sichten sind virtuelle Relationen und können Joins über mehrere Tabellen bilden, wobei nicht jedes Attribut, was zur Erstellung der Sicht genutzt wurde, auch in der Ergebnisrelation enthalten sein muss.

Auf Seite der Anonymisierung wurden Techniken betrachtet, die spalten- oder auch zeilenweise vorgehen. Damit soll sichergestellt werden, dass eine möglichst geringe Modifikation der Daten vorgenommen wird, allerdings trotzdem so viele Informationen entfernt werden, wie nötig, um den entsprechenden Grad der Anonymisierung zu erfüllen. Wenn zu viele Informationen entfernt werden, können die Daten ansonsten sehr schnell uninteressant werden, da keine Auswertungen mehr möglich sind.

Da eine komplette Automatisierung eines Angriffs so gut wie unmöglich ist, wurde ein Programm entwickelt, welches es einem Angreifer erlaubt, detailliertere Informationen über die vorliegenden Daten zu finden und so einfacher einen passenden Angriffsvektor wählen zu können. Hierfür werden dem Nutzer

Statistiken zu den Informationen in der Ergebnisrelation präsentiert und es ist auch möglich, die zurück gelieferten Werte einzuschränken, um fehlerhafte Daten auszuschließen.

In einer Fortsetzung des Programms könnte eine Möglichkeit geschaffen werden, Ergebnisse aus Datenbankanfragen aus lokalen Dateien, zum Beispiel Comma-separated values (csv)-Dateien, einlesen zu können. Dadurch kann auch eine nur als Datei verfügbare Tabelle analysiert werden. Derzeit ist nur die Möglichkeit gegeben, ein ResultSet einer Datenbankabfrage zu bewerten. Weiterhin wäre eine „Generalisierungsdatenbank“ wünschenswert. Darin sollten Informationen gespeichert werden, wie sich verschiedene Attribute generalisieren lassen. Damit könnte das Programm auch für eine Homogenitätsattacke genutzt werden. Hierbei ist allerdings zu beachten, dass dies sehr viel Aufwand bedeutet, da die Benennung von Spalten meist sehr individuell ist und auch die Schreibweise und Wahl der einzelnen Attributwerte. Selbst wenn eine Generalisierung für ein Attribut vorliegt, könnte es daher schwierig sein, automatisiert die richtige Spalte zu finden und so eine Generalisierung der Werte vorzunehmen. Es wäre auch denkbar, die Bereichseinschränkung für verschiedene Werte weiter zu verbessern und beispielsweise fehlerhafte Werte automatisiert zu erkennen.

Weiterhin ist es in Bezug auf das PArADISE-Framework denkbar die „Answering-Queries-Using-Views“-Technik zu nutzen, um die Anfragen, welche durch die Cloud-Schicht beantwortet werden, umzusetzen. Es wäre damit möglich, den Zugang zur Datenbank deutlich einzuschränken und eine Art des Whitelistings für Anfragen, in Form der Sichten, zu verwirklichen. Es muss allerdings vorab geprüft werden, ob sich mit den Sichten nicht doch Informationen extrahieren lassen, die eigentlich nicht gestattet sein sollten. Dies ist besonders wichtig, wenn mehr als eine Sicht zur Verfügung stehen. Hier muss geprüft werden, inwiefern sich die verfügbaren Sichten zu bösartigen Anfragen kombinieren lassen. Es ist selbstverständlich auch nötig, die Ergebnisrelationen der Sichten zu anonymisieren, sollten die Anonymisierungsmaße nicht von allein erfüllt sein.

Die AQuV-Technik muss erweitert werden, da bisher nur einfache Anfrageoperationen in Anfragen und Sichten erlaubt sind, das Anwendungsszenario aber komplexe Aggregate, Gruppierungen etc. erfordert. Darauf sind die Techniken bisher noch nicht ausgelegt.

Literaturverzeichnis

[Boc09] Bockermann, Christian and Apel, Martin and Meier, Michael: *Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling*, Seiten 196–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Com16] ComputerBase: *Datenschützer warnen vor Google Home.* `https://www.computerbase.de/2016-05/intelligente-sprachassistenten-datenschuetzer-warnen-vor-google-home/`, 2016. Erscheinungsdatum: 26.05.2016, Zuletzt aufgerufen am 10.10.2016.

[DHI12] Doan, AnHai, Alon Halevy und Zachary Ives: *Principles of Data Integration.* Seiten 21–63. Morgan Kaufmann, Elsevier, Inc., 2012.

[Dom08] Domingo-Ferrer, Josep and Torra, Vicenç: *A critique of k-anonymity and some of its enhancements.* In: *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, Seiten 990–993. IEEE, 2008.

[Dwo08] Dwork, Cynthia: *Differential privacy: A survey of results.* In: *International Conference on Theory and Applications of Models of Computation*, Seiten 1–19. Springer, 2008.

[Dwo09] Dwork, Cynthia and Smith, Adam: *Differential Privacy and for Statistics: What we Know and What we Want to Learn.* In: *Journal of Privacy and Confidentiality*, Band 1, Seiten 135–154, 2009.

[Eck16a] Eckert, Svea and Klofta, Jasmin: *Nackt im Netz: Millionen Nutzer ausgespäht.* `https://www.ndr.de/nachrichten/netzwelt/Nackt-im-Netz-Millionen-Nutzer-ausgespaeht,nacktimnetz100.html`, 2016. Erscheinungsdatum: 01.11.2016, Zuletzt aufgerufen am 02.11.2016.

[Eck16b] Eckert, Svea and Klofta, Jasmin and Strozyk, Jan Lukas: *Nackt im Netz: Auch initime Details von Bundespolitikern im Handel.* `http://daserste.ndr.de/panorama/archiv/2016/Nackt-im-Netz-Intime-Details-von-Politikern-im-Handel,nacktimnetz110.html`, 2016. Erscheinungsdatum: 03.11.2016, Zuletzt aufgerufen am 04.11.2016.

[Fri10] Friedman, Arik and Schuster, Assaf: *Data mining with differential privacy.* In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 493–502. ACM, 2010.

[GH14] Grunert, Hannes und Andreas Heuer: *Big Data und der Fluch der Dimensionalität.* In: *Proceedings of the 26th GI-Workshop Grundlagen von Datenbanken, Bozen-Bolzano, Italy, October 21st to 24th, 2014*, Seiten 29–34, 2014.

[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, 2015.

[GKB+16] Grunert, Hannes, Martin Kasparick, Björn Butzin, Andreas Heuer und Dirk Timmermann: *From Cloud to Fog and Sunny Sensors.* In: Krestel, Ralf, Davide Mottin und Emmanuel Müller (Herausgeber): *Proceedings of the Conference „Lernen, Wissen, Daten, Analysen“, Potsdam, Germany, September 12-14, 2016.*, Band 1670 der Reihe *CEUR Workshop Proceedings*, Seiten 83–88. CEUR-WS.org, 2016.

[Gru16] Grunert, Hannes and Heuer, Andreas: *Datenschutz im PArADISE.* Datenbank-Spektrum, 16(2):107–117, 2016.

[Hal01] Halevy, Alon Y: *Answering queries using views: A survey.* The VLDB Journal, 10(4):270–294, 2001.

[Hau07] Hauf, Dietmar: *Allgemeine Konzepte - K-Anonymity, l-Diversity and T-Closeness.* `https://dbis.ipd.kit.edu/img/content/SS07Hauf_kAnonym.pdf`, 2007. Zuletzt aufgerufen am 14.10.2016.

[Heu15] Heuser, Uwe Jean: *Datenanalyse - Wir werden vermessen.* `http://www.zeit.de/2015/07/datenanalyse-mensch-verhalten-vorhersage-freiheit`, Februar 2015. Zuletzt aufgerufen am 26.09.2016.

[Hur16] Hurtz, Simon: *Whatsapp teilt Telefonnummern mit Facebook - das können Sie tun.* `http://www.sueddeutsche.de/digital/2.220/messenger-whatsapp-teilt-telefonnummern-mit-facebook-das-koennen-sie-tun-1.3172711`, 2016. Erscheinungsdatum: 25.09.2016, Zuletzt aufgerufen am 03.11.2016.

[Kam08] Kamra, Ashish and Terzi, Evimaria and Bertino, Elisa: *Detecting anomalous access patterns in relational databases.* The VLDB Journal, 17(5):1063–1077, 2008.

[Kun10] Kundu, Amlan and Sural, Shamik and Majumdar, A. K.: *Database intrusion detection using sequence alignment.* International Journal of Information Security, 9(3):179–191, 2010.

[Lee01] Lee, Wenke and Xiang, Dong: *Information-theoretic measures for anomaly detection.* In: *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, Seiten 130–143. IEEE, 2001.

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

[LRO96] Levy, A., A. Rajaraman und J. Ordille: *Querying Heterogeneous Information Sources Using Source Descriptions.* Technical Report 1996-61, Stanford InfoLab, 1996.

[LSS00] Lee, Victor C. S., John A. Stankovic und Sang Hyuk Son: *Intrusion Detection in Real-Time Database Systems via Time Signatures.* In: *Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium, RTAS 2000, Washington, DC, USA, May 31 - June 2, 2000*, Seiten 124–133, 2000.

[Mat10] Mathew, Sunu and Petropoulos, Michalis and Ngo, Hung Q. and Upadhyaya, Shambhu: *A Data-Centric Approach to Insider Attack Detection in Database Systems*, Seiten 382–401. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[MKGV07] Machanavajjhala, Ashwin, Daniel Kifer, Johannes Gehrke und Muthuramakrishnan Venkitasubramaniam: *L-diversity: Privacy Beyond K-anonymity.* ACM Trans. Knowl. Discov. Data, 1(1), März 2007.

[MX07] Motwani, Rajeev und Ying Xu: *Efficient algorithms for masking and finding quasi-identifiers.* In: *Proceedings of the Conference on Very Large Data Bases (VLDB)*, Seiten 83–93, 2007.

[MyS] MySQL: *Employee Sample Database Version 1.0.6.* `https://launchpad.net/test-db/employees-db-1/1.0.6/+download/employees_db-full-1.0.6.tar.bz2`. Version 1.0.6, Zuletzt aufgerufen am 05.12.2016.

[PH01] Pottinger, Rachel und Alon Y. Halevy: *MiniCon: A scalable algorithm for answering queries using views.* VLDB J., 10(2-3):182–198, 2001.

[Sam98] Samarati, Pierangela and Sweeney, Latanya: *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.* Technischer Bericht, Technical report, SRI International, 1998.

[San14] Santos, Ricardo Jorge and Bernardino, Jorge and Vieira, Marco: *Approaches and Challenges in Database Intrusion Detection.* ACM SIGMOD Record, 43(3):36–47, 2014.

[Sha10] Sha, Chaofeng and Li, Yi and Zhou, Aoying: *On t-closeness with KL-divergence and semantic privacy.* In: *International Conference on Database Systems for Advanced Applications*, Seiten 153–167. Springer, 2010.

[Sri06] Srivastava, Abhinav and Sural, Shamik and Majumdar, Arun K: *Database intrusion detection using weighted sequence mining.* Journal of Computers, 1(4):8–17, 2006.

[Swe02] Sweeney, Latanya: *k-anonymity: a model for protecting privacy.* International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(05):557–570, 2002.

[UCI96] UCI Machine Learning Repository: *Adult Data Set.* `http://archive.ics.uci.edu/ml/datasets/Adult`, 1996. Zuletzt aufgerufen am 27.09.2016.

A. Datenträger mit Software

Selbständigkeitserklärung zur Bachelorarbeit

Ich, Johannes Goltz, erkläre, dass ich die vorliegende Arbeit zum Thema „*De-Anonymisierungsverfahren: Kategorisierung und deren Anwendung für Datenbankanfragen*“ selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, 06. Februar 2017	
Ort, Datum	Unterschrift