technische universität
dortmund

Diplomarbeit

Untersuchungen zum Inferenzschutz von fragmentierten Speicherungen von Datenbankinstanzen

Marcel Preuß

30. September 2010

Betreuer:
Prof. Dr. Joachim Biskup
Dr. Lena Wiese

Fakultät für Informatik
Informationssysteme und Sicherheit (Ls6)
Technische Universität Dortmund
http://ls6-www.cs.tu-dortmund.de
preuss@ls6.cs.tu-dortmund.de

Was wir wissen, ist ein Tropfen.
Was wir nicht wissen, ein Ozean.

Sir Isaac Newton

Diese Diplomarbeit wurde mit LaTeX gesetzt.

Danksagung

An dieser Stelle meiner Diplomarbeit möchte ich die Gelegenheit nutzen, mich bei den Menschen zu bedanken, die mich bei der Ausgestaltung dieser Diplomarbeit entscheidend unterstützt haben.

Dabei möchte ich an erster Stelle ganz besonders Herrn Prof. Dr. Joachim Biskup danken, der mich über den kompletten Bearbeitungszeitrum der Diplomarbeit hervorragend betreut hat. Dabei waren mir vor allem seine weitsichtigen Vorschläge, in welche Richtung der jeweils aktuelle Stand der Diplomarbeit weiter entwickelt werden könnte, stets eine große Hilfe. Zudem hat er mir die Möglichkeit gegeben, im Rahmen meines Studiums der Informatik über vier Jahre lang als Studentische Hilfskraft an seinem Lehrstuhl zu arbeiten. Dadurch konnte ich bereits während meines Studiums Einblicke in den Alltag des wissenschaftlichen Arbeitens erlangen. Ganz besonders erfreut es mich, dass wir diese Kooperation aufrecht erhalten und ich nach Abschluss meines Studiums die Tätigkeit als Wissenschaftlicher Mitarbeiter an seinem Lehrstuhl aufnehmen kann.

Ebenfalls möchte ich mich auch ganz herzlich bei Frau Dr. Lena Wiese bedanken, die mir bei der Ausgestaltung meiner Diplomarbeit zur Seite stand. Dabei waren mir ihre Erfahrungen aus der wissenschaftlichen Praxis stets eine große Hilfe, so dass auch sie zum Gelingen dieser Diplomarbeit entscheidend beigetragen hat. Für den neuen Lebensabschnitt, der für sie und ihre Familie nun mit dem Umzug nach Japan beginnt, möchte ich ihr und ihrer Familie an dieser Stelle sowohl beruflich als auch privat alles Gute und viel Erfolg wünschen.

Zudem möchte ich auch nicht vergessen, mich bei meinen Eltern zu bedanken, die mich während des kompletten Zeitraums meines Studiums finanziell unterstützt haben. Neben der finanziellen Unterstützung war sicherlich auch – insbesondere während der Prüfungsphasen – die ideelle Unterstützung eine große Hilfe für mich.

Zusammenfassung

In der heutigen Informationsgesellschaft hat Information den Stellenwert einer zentralen Ressource eingenommen, die es zu schützen gilt. Um in Informationssystemen gespeicherte Information vor nicht autorisierten Zugriffen zu schützen, werden häufig formale Vertraulichkeitsanforderungen definiert, die durch das verwendete System automatisiert durchgesetzt werden sollen. Oft sind dabei die einzelnen Informations-Aspekte, die in einem Informationssystem verwaltet werden, für sich betrachtet jeweils nicht besonders schützenswert. Dafür sind Assoziationen zwischen verschiedenen Informations-Aspekten, die selbst wiederum als Information aufgefasst werden können, schützenswert.

Zum Schutz sensibler Assoziationen wird von einigen Autoren vorgeschlagen, Datenbankinstanzen gemäß bestimmter Vertraulichkeitsanforderungen durch das Konzept der sogenannten Fragmentierung in verschiedene Teile zu zerlegen. Dazu existieren verschiedene Ansätze, für die von den jeweiligen Autoren erläutert wird, wie jeweils unmittelbare, unautorisierte Zugriffe auf vertrauliche Information unterbunden werden. Dabei wird aber nicht darauf eingegangen, ob die jeweils zum Einsatz kommenden Ansätze zur Fragmentierung einer Datenbankinstanz auch Schutz vor Inferenzen bieten, durch die sensible Information auf Basis der Kenntnis nicht sensibler Information erschlossen werden kann.

Im Gegensatz dazu ist für die Verfahren der sogenannten kontrollierten Anfrageauswertung die Eigenschaft der Inferenzsicherheit formal nachgewiesen. In dieser Diplomarbeit wird ein logik-orientiertes Framework der kontrollierten Anfrageauswertung entwickelt, in dem die Ansätze zur Fragmentierung von Datenbankinstanzen modelliert werden können. Innerhalb einer solchen Modellierung wird für eines der existierenden Verfahren zur Fragmentierung von Datenbankinstanzen konkret nachgewiesen, dass dieses Verfahren sicher vor unerwünschten Inferenzen ist, sofern davon ausgegangen werden kann, dass ein Benutzer nicht über explizites Vorwissen über das Informationssystem verfügt oder dieses Vorwissen ausschließlich aus einer eingeschränkten Menge funktionaler Abhängigkeiten besteht.

Abstract

In these days information has become one of the most important resources, which has to be protected. In order to protect information from undesired accesses of unauthorized users confidentiality requirements are defined by setting up a confidentiality policy. According to such a confidentiality policy a system should enforce the defined confidentiality requirements autonomously. Often, single pieces of information are not seen as confidential from an isolated point of view. In contrast to that associations between different pieces of information, which can also be interpreted as a kind of information, are seen as confidential and have to be protected.

To achieve this protection of sensitive associations, some authors suggest to split a database instance into several pieces by using the so called concept of fragmentation. There are several different concepts based on fragmentation and for each of these concepts the corresponding authors describe how unauthorized (direct) accesses to confidential information are prohibited. But in this context it is not shown that confidential information cannot be inferred by employing inferences, which may offer the possibility to infer confidential information based on the knowledge of information that is not declared as confidential.

In contrast to that there are several concepts of so called controlled query evaluation and for each of these concepts it is proven that a defined confidentiality policy is enforced by prohibiting such inferences. In this diploma thesis a logic-oriented framework, which allows modelling the concepts of fragmentation of database instances within the framework of controlled query evaluation, is developed. For one of the proposed concepts of fragmentation of database instances such a modelling is used to prove that harmful inferences are not possible, if confidentiality requirements are enforced by using this specific concept of fragmentation and a user either does not have any explicit a priori knowledge or just has a priori knowledge about a restricted class of functional dependencies.

Inhaltsverzeichnis

1 Einleitung

Bedingt durch den Prozess des gesellschaftlichen Wandels hin zur Dienstleistungsgesellschaft hat Information heute den Stellenwert einer zentralen Ressource erlangt. Deshalb kann man im Kontext der Dienstleistungsgesellschaft auch von einer Informationsgesellschaft sprechen, in der die Verwaltung von Information als zentrale Herausforderung angesehen werden kann.

1.1 Herausforderungen der Informationsgesellschaft

Eine zentrale Eigenschaft des immateriellen Begriffs der Information besteht darin, dass der Austausch von Information möglich ist. In der Regel kann gerade dem Austausch von Information eine zentrale Bedeutung zugemessen werden, da Information insbesondere dann sinnvoll eingesetzt werden kann, wenn sie (unter Umständen auch von Dritten) verarbeitet wird. So ist es beispielsweise in der Wirtschaft unerlässlich, dass projektbezogene Information zwischen kooperierenden Unternehmen ausgetauscht wird, und innerhalb des Managements eines Unternehmens muss Information über die zukünftige strategische Ausrichtung des Unternehmens am Markt ausgetauscht werden.

Um den Austausch von Information möglichst effizient zu gestalten, ist man üblicherweise bestrebt, diesen automatisiert mit Hilfe von Informationssystemen durchzuführen. Dabei kristallisiert sich als ein zentrales Problem heraus, wer Zugriff auf welche Information bekommen darf. So gilt im obigen Beispiel die Information zur zukünftigen strategischen Ausrichtung des Unternehmens in aller Regel als streng vertraulich und darf nur von dazu autorisierten Personen abgerufen werden.

Da durch Informationssysteme die Verwaltung von Information weitestgehend automatisiert werden kann, wird es sowohl privatwirtschaftlich geführten Unternehmen als auch staatlichen Organisationen stark vereinfacht, immer größere Mengen an Information zu sammeln. Dabei handelt es sich oft auch um personenbezogene Daten, deren Speicherung aufgrund gesetzlicher Vorschriften nur dann erlaubt ist, wenn gewisse Anforderungen zum Schutz der Information, die in diesen Daten repräsentiert wird, gewährleistet sind. So ist es beispielsweise im Gesundheitswesen

unstrittig, dass zwar eine Notwendigkeit zur Speicherung von Krankenakten besteht, diese aber an strenge Sicherheitsvorkehrungen gebunden sein muss, um einen Missbrauch dieser sensiblen personenbezogenen Daten zu verhindern.

Diese vorgestellten Beispiele motivieren deutlich, dass dem Gebiet der Sicherheit in der Informationstechnologie, das sich unter anderem mit dem Schutz von Daten und der in ihnen repräsentierten Information beschäftigt, in unserer heutigen Informationsgesellschaft mehr denn je ein besonderer Stellenwert zukommen muss.

1.1.1 Sicherheit in Informationssystemen

Im Folgenden soll erst einmal der oben motivierte Begriff der Sicherheit konkretisiert werden, indem auf Schutzziele, die durch Sicherheitsmechanismen gewährleistet werden sollen, eingegangen wird. Dabei sind vor allem die nachfolgend erläuterten Schutzziele von vorrangiger Bedeutung [28, Kap. 1.2], [6, Kap. 1.1]:

Verfügbarkeit: Die Funktionalität des Systems soll sichergestellt sein. Das heißt, ein autorisierter (und authentifizierter) Benutzer soll auf ein für ihn freigegebenes Datum unter Beachtung der für ihn auf diesem Datum erlaubten Operationen (z.B. Schreiben, Lesen) Zugriff erhalten können.

Vertraulichkeit: Jeglicher unautorisierter Informationsgewinn eines Benutzers (z.B. durch lesenden Zugriff eines Benutzers auf für ihn nicht freigegebene Daten) soll durch das System unterbunden werden. Dazu müssen Anfragen an das System, die einen solchen unautorisierten Informationsgewinn ermöglichen würden, geeignet behandelt (z.B. abgelehnt) werden.

Integrität: Der Informationsgehalt des Systems soll vor unerlaubten Manipulationen geschützt werden. Das heißt, dass schreibende Zugriffe auf schützenswerte Daten durch dazu nicht autorisierte Benutzer unterbunden werden müssen.

Authentizität: Die Identität eines Benutzers soll durch geeignete charakterisierende Eigenschaften (z.B. Passwörter, biometrische Merkmale oder kryptographische Schlüssel) überprüft werden. Dadurch soll sichergestellt werden, dass im Kontext der Integrität die Quellen von Informationsflüssen glaubwürdigen Benutzern und im Kontext der Vertraulichkeit die Ziele von Informationsflüssen vertrauenswürdigen Benutzern entsprechen.

Bezogen auf die oben genannten Schutzziele ist auffällig, dass die Kombination der Anforderungen, die jeweils aus den Schutzzielen der Verfügbarkeit und der Vertraulichkeit heraus resultieren, konfliktbehaftet sein kann. So führt maximale Verfügbarkeit im Extremfall dazu, dass jeder Benutzer Zugriff auf alle Daten des Systems erhalten kann und Vertraulichkeits-Eigenschaften missachtet werden. Im Gegensatz

dazu kann ein maximal vertrauliches System überhaupt keine Zugriffe auf Daten zulassen, so dass die gemäß der Verfügbarkeit gewünschte Funktionalität des Systems nicht bereitgestellt werden kann. Aufgrund dessen ist es in der Regel notwendig, einen für das jeweilige System geeigneten Kompromiss zwischen den Schutzzielen der Verfügbarkeit und der Vertraulichkeit zu finden.

Zur Wahrung von Vertraulichkeitsanforderungen ist es notwendig, dass die entsprechenden Schutzmechanismen eines Systems präventiv arbeiten [30, Kap. 2.1.1]. Aufgrund der oben diskutierten Eigenschaft, dass Information immateriell und kopierbar ist, kann einem Benutzer einmal gewonnene Information zu einem späteren Zeitpunkt nicht wieder entzogen werden. Deshalb ist es entscheidend, dass Anfragen eines Benutzers vor Beantwortung dieser auf Einhaltung der geforderten Vertraulichkeitsanforderungen hin überprüft werden. Im oben genannten Beispiel des Wirtschafts-Unternehmens könnten beispielsweise die vertraulichen Daten zur zukünftigen strategischen Ausrichtung des Unternehmens von einem Konkurrenten auch nach der Aufdeckung, dass dieser Konkurrent Kenntnis von dieser Information erlangt hat, genutzt werden, um die Position dieses Unternehmens am Markt zu schwächen. Ein nachträgliches Aufdecken von unautorisiertem Informationsgewinn kann aber in vielen Fällen dennoch sinnvoll sein, um auf die jeweilige Situation angepasste Strategien zur Schadensbegrenzung entwickeln zu können.

1.1.2 Präventive Umsetzung sicherheitsrelevanter Schutzziele

Um Vertraulichkeit und Integrität von Daten präventiv zu schützen, existiert das Konzept der Zugriffskontrolle. Dabei gibt es im Wesentlichen die Konzepte der sogenannten diskretionären Zugriffskontrolle und der sogenannten mandatorischen Zugriffskontrolle [6, Kap. 4.4].

Das Konzept der diskretionären Zugriffskontrolle ist von der grundsätzlichen Idee geprägt, dass ein Benutzer des Systems für die von ihm verwalteten Daten jeweils selbst die gewünschte Menge an Regeln zur Zugriffskontrolle verwaltet. Eine Zugriffsregel, die im Wesentlichen festlegt, welcher Benutzer auf welches Datum mit welcher Operation (z.B. Lesen, Schreiben) zugreifen darf, kann dabei im einfachsten Fall aus einem entsprechenden expliziten Tripel bestehen. Je nach Mächtigkeit der Sprache zur Definition des Regelwerks können solche Zugriffsregeln aber auch abstrakt definiert werden und aus einer solchen abstrakten Beschreibung des Regelwerks (z.B. auf Basis von Logik) implizit abgeleitet werden [9, Kap. 9.1].

Bei der mandatorischen Zugriffskontrolle wird im Hinblick auf die Kontrolle von Informationsflüssen durch einen Administrator ein zentrales Regelwerk verwaltet, das die Zugriffsrechte von Benutzern auf Daten festlegt. Dabei werden auf Basis

einer endlichen Menge partiell geordneter Klassifikationen Daten klassifiziert und Benutzer erhalten Freigaben aus dieser Menge von Klassifikationen [9, Kap. 9.8]. Zur Wahrung der Vertraulichkeit gilt es, Information ausschließlich in Richtung höherer oder gleich hoher und damit nicht weniger vertrauenswürdiger Klassifikationen fließen zu lassen. Im Falle eines lesenden Zugriffs auf ein Datum muss die Freigabe des anfragenden Benutzers, der als Ziel dieses Informationsflusses fungiert, also mindestens so hoch wie die Klassifikation des angeforderten Datums, welches die Quelle dieses Informationsflusses ist, sein, damit Vertraulichkeit gewahrt bleibt. Bei einem schreibenden Zugriff entspricht der Benutzer hingegen der Quelle des Informationsflusses und darf dementsprechend zur Wahrung der Vertraulichkeit keine Freigabe haben, die höher als die Klassifikation des zu schreibenden Datums ist. Zur Wahrung der Integrität muss bei der mandatorischen Zugriffskontrolle ein dualer Ansatz zu den Überlegungen zur Wahrung der Vertraulichkeit verfolgt werden. Diese Problematik wird in [9, Kap. 9.8.3] behandelt.

Die oben angesprochenen Mechanismen zur Zugriffskontrolle sind dazu geeignet, den Zugriff von Benutzern auf einzelne Daten zu kontrollieren. Dabei wird von dem Zugriffskontroll-Mechanismus des Systems für jede einzelne Operation, die ein authentifizierter Benutzer auf ein bestimmtes Datum anwenden möchte, überprüft, ob diese Operation gemäß des vorhandenen Regelwerks zur Zugriffskontrolle erlaubt ist oder nicht. Entsprechend dieser Entscheidung wird die vom Benutzer gestellte Anfrage an das System entweder gestattet oder zurückgewiesen. An diesem Verfahren kann sich aber als problematisch erweisen, dass jede Anfrage eines Benutzers isoliert für sich entschieden wird, ohne diese in den Kontext bisheriger Anfragen dieses Benutzers einzuordnen.

Dadurch existiert die Gefahr, dass ein Benutzer aus der Kombination verschiedener (prinzipiell unkritischer) Anfragen heraus Schlussfolgerungen ziehen kann, deren Informationsgehalt ihm eigentlich vorenthalten bleiben sollte. Gerade in Informationssystemen, in denen Daten strukturiert gespeichert werden und in der Regel einen semantischen Bezug zueinander aufweisen, ist die Gefahr solcher Inferenzen, die das Erschließen vertraulicher Information aus nicht vertraulicher Information heraus bezeichnen, gegeben [21, Kap. 1.3.2]. Dabei sollte auch in Betracht gezogen werden, dass ein Benutzer zum Erschließen von vertraulicher Information auch sein Vorwissen über generelle Sachverhalte oder das System nutzen kann. Eine Übersicht über verschiedene Ausprägungen des Inferenzproblems ist in [29] zu finden.

Als Beispiel dazu soll das Informationssystem eines Krankenhauses betrachtet werden, in dem für jeden Patienten die behandelte Krankheit und die zur Behandlung verwendeten Medikamente verwaltet werden. Dabei hat zwar das komplette medizinische Personal Zugriff darauf, welcher Patient welche Medikamente bekommt, der Zugriff auf die konkreten Krankheiten der Patienten soll aber nur Ärzten gestattet

sein und dem restlichen Pflegepersonal vorenthalten bleiben. Auf Basis des Hintergrundwissens, welche Kombination von Medikamenten zur Behandlung welcher Krankheiten genutzt wird, kann die bei einem Patienten behandelte Krankheit unter Umständen aber auch von dazu nicht autorisiertem Pflegepersonal erschlossen werden. Wenn die Schnittmenge der Krankheiten, die mit den verabreichten Medikamenten behandelt werden können, einelementig ist, kann die Krankheit eines Patienten eindeutig erschlossen werden. Andernfalls besteht die Möglichkeit, dass die Menge möglicher Krankheiten zumindest stark eingegrenzt werden kann. Auch dadurch kann ein nicht unerheblicher Informationsgewinn entstehen.

1.1.3 Externe Speicherung von Datenbankinstanzen

Neben den bereits diskutierten Anforderungen bezüglich der Sicherheit von Information existiert in unserer wirtschaftlich geprägten Gesellschaft auch der Wunsch, Daten möglichst kostengünstig speichern zu können. Zur persistenten Speicherung von Daten innerhalb des eigenen Unternehmens müssen geeignete Rechenzentren eingerichtet werden, um Datenbanksysteme betreiben zu können. Dies erfordert die Bereitstellung von Räumlichkeiten und die Anschaffung entsprechender Hard- und Software. Des Weiteren müssen Fachkräfte eingestellt werden, die mit der Administration und der Wartung der Systeme betraut werden. All diese Aspekte sind betriebswirtschaftlich gesehen mit Kosten verbunden.

Zudem muss sichergestellt werden, dass die bereitgestellten Datenbanksysteme möglichst ständig verfügbar sind, da der Ausfall dieser in vielen Fällen mit Einbußen der Produktivität des Unternehmens verbunden ist. Oft führen aber gerade notwendige Wartungsarbeiten an den Systemen oder administrative Eingriffe wie Updates auf neue Software-Versionen zu Ausfallzeiten.

Deshalb gibt es spezialisierte Dienstleister, deren Geschäftsmodell darin besteht, Datenbanksysteme zu betreiben und diese an andere Unternehmen zu vermieten. Die Kunden eines solchen Dienstleisters erhalten dann per Internet Zugriff auf die angemieteten Datenbanksysteme und können oft Einsparungen verbuchen, weil der Dienstleister das Datenbanksystem aufgrund seiner Spezialisierung kostengünstiger betreiben kann als das anmietende Unternehmen selbst. Zudem können durch standardisierte Prozesse oft Ausfallzeiten minimiert werden. Dieses hier vorgestellte Szenario wird auch als „Database as a Service“-Paradigma bezeichnet [31].

Problematisch an diesem Szenario ist aus Sicht des anmietenden Unternehmens allerdings, dass der eigene Datenbestand in den Verwaltungsbereich des fremden Dienstleisters gegeben wird, so dass dieser ohne zusätzliche Schutzmechanismen

Zugriff auf den kompletten Datenbestand bekommen würde. Aufgrund der in Kapitel 1.1 diskutierten Bedeutung von Datenbeständen müssen deshalb Überlegungen getätigt werden, wie diese ausgelagerten Daten geschützt werden können.

1.2 Ziele dieser Diplomarbeit

Wie in Kapitel 1.1.3 dargelegt, kann es aus wirtschaftlichen Erwägungen sinnvoll sein, Datenbanksysteme nach dem „Database as a Service“-Paradigma auswärtig zu verwalten. Dabei müssen aber Maßnahmen zum Schutz der Vertraulichkeit der Daten ergriffen werden. Zu diesem Zweck gibt es unter anderem den Vorschlag der fragmentierten Speicherung von Datenbankinstanzen (siehe Kapitel 3). Bei diesem Konzept wird eine Datenbankinstanz derart in einzelne Fragment-Instanzen zerlegt, dass keine dieser Fragment-Instanzen für sich betrachtet unmittelbaren Zugriff auf vertrauliche Information oder auch vertrauliche Assoziationen zwischen verschiedenen Aspekten der repräsentierten Information erlaubt.

Fraglich ist aber, ob dieser Schutz vor unmittelbarem Zugriff auf vertrauliche Information auch potentiell mögliche Inferenzen unterbindet. Wie bereits einführend in Kapitel 1.1.2 erläutert ist es oft trotz Verfahren zum Schutz vor unautorisierten Zugriffen möglich, aus der Kombination mehrerer Aspekte nicht vertraulicher Information (und möglicherweise weiterem Hintergrundwissen) unerwünschte Rückschlüsse auf eigentlich vertrauliche Information ziehen zu können. In Datenbanksystemen bietet gerade die strukturierte Speicherung von Daten, bei der semantische Bezüge zwischen verschiedenen Aspekten der repräsentierten Information berücksichtigt werden, einen möglichen Ansatzpunkt für solche Inferenzen.

In Form der sogenannten kontrollierten Anfrageauswertung (siehe Kapitel 4) existiert aber ein Verfahren, das unerwünschte Inferenzen beweisbar unterbindet, indem der mögliche Informationsgewinn eines Benutzers beschränkt wird. Dabei wird bei jeder Anfrage an das System überprüft, ob eine korrekte Antwort einem rational denkenden Benutzer das Erschließen von vertraulicher Information ermöglichen würde. Ziel ist es dabei, den für einen Benutzer möglichen Informationsgewinn so weit zu begrenzen, dass für diesen Benutzer aus rationalen Erwägungen heraus stets eine alternative Sicht auf den Datenbestand des Systems existiert, in der die als vertraulich deklarierte Information nicht notwendigerweise gelten muss.

Ziel dieser Diplomarbeit ist es, zu untersuchen, inwieweit die bei der kontrollierten Anfrageauswertung garantierten Vertraulichkeitseigenschaften auch im Kontext der fragmentierten Speicherung von Datenbankinstanzen erreicht werden.

2 Grundlagen der Datenbanktheorie

Für die Ansätze, die in den nachfolgenden Kapiteln vorgestellt werden, ist es wichtig, eine strukturierte Sichtweise auf Datenbestände zu haben, auf der die dort vorgestellten Ansätze aufgebaut werden können. Eine solche strukturierte Sichtweise bietet das relationale Datenmodell aus dem Bereich der Datenbanktheorie.

2.1 Das relationale Datenmodell

In diesem Kapitel soll in das relationale Datenmodell nach der von Biskup vorgestellten Notation eingeführt werden (siehe [7, Kap. 8.1]). Als Basis für die nachfolgenden Definitionen des relationalen Datenmodells wird hier davon ausgegangen, dass mit $\mathcal{R}$ die Menge aller möglichen Relationensymbole und mit $\mathcal{A}$ die unendliche Menge aller Attribute bezeichnet wird. Dabei ist jedes Attribut von einem bestimmten Datentyp aus der Menge $\mathcal{B}$ der semantischen Bereichsnamen. Der einem Attribut $a \in \mathcal{A}$ zugeordnete semantische Bereichsname legt fest, welche Werte dieses Attribut a annehmen kann.

Um festzulegen, welche konkreten Werte zu einem semantischen Bereichsnamen gehören, ordnet die als global bekannt angenommene Funktion $\beta : \mathcal{B} \to \wp(\mathcal{K})$ jedem semantischen Bereichsnamen eine Teilmenge der unendlichen Menge der Konstantenzeichen $\mathcal{K}$ zu. Eine solche Teilmenge kann beispielsweise aus den (unendlich vielen) Symbolen für die Elemente der Menge $\mathbb{N}$ der natürlichen Zahlen oder aus allen (möglicherweise unendlich vielen) Wörtern, die sich aus den Buchstaben eines bestimmten Alphabets bilden lassen, bestehen.

2.1.1 Relationenschema und Relationeninstanz

Auf Basis der oben definierten Grundlagen kann nun das Konzept des sogenannten Relationenschemas vorgestellt werden, das einer Art Schablone für zu repräsentierende Information entspricht. Auf diesem Konzept kann später die Definition der Relationeninstanz aufgebaut werden.

Definition 2.1 (Relationenschema) *Sei $\mathcal{R}$ die Menge aller möglichen Relationensymbole und $\mathcal{A}$ die unendliche Menge aller Attribute. Dann ist $\langle R|A_R|SC_R\rangle$ ein Relationenschema mit Relationensymbol $R \in \mathcal{R}$ und der endlichen Menge $A_R \subset \mathcal{A}$ an Attributen. Die Menge SC_R enthält die lokalen semantischen Bedingungen des Relationenschemas, in denen nur das Relationensymbol R selbst und die Gleichheitsrelation vorkommen dürfen.*

Ein Relationenschema $\langle R|A_R|SC_R\rangle$ legt also im Wesentlichen fest, welche Struktur Daten haben sollen, die nach diesem Schema gespeichert werden. Dazu wird eine Menge A_R von Attributen festgelegt, die in dieser spezifischen Kombination dazu geeignet sind, den Typ Information, der nach diesem Schema repräsentiert werden soll, zu beschreiben. Zudem können die lokalen semantischen Bedingungen der Menge SC_R die Werte, mit denen in einer beliebigen Relationeninstanz zu diesem Schema (siehe unten) ein bestimmtes Attribut in einem bestimmten Tupel belegt werden kann, in Abhängigkeit von anderen Attributwerten dieser Relationeninstanz einschränken. Derartige lokale semantische Bedingungen sind beispielsweise funktionale Abhängigkeiten (siehe Kapitel 2.2).

Auf Basis der Definition eines Relationenschemas können Relationeninstanzen definiert werden. Während ein Relationenschema einer Art Schablone für die Form der Repräsentation von Information entspricht, ist eine Relationeninstanz eine Ausprägung zu einem Schema, in der konkrete Information in der durch das Schema vorgegebenen Form enthalten ist.

Definition 2.2 (Relationeninstanz) *Sei $\langle R|A_R|SC_R\rangle$ ein Relationenschema gemäß Definition 2.1. Eine Relationeninstanz r über dem Universum d ist genau dann eine gültige Ausprägung zu Schema $\langle R|A_R|SC_R\rangle$, wenn*

(i) d eine endliche Teilmenge der Konstantenzeichen $\mathcal{K}$ ist,

(ii) r eine endliche Teilmenge aus der Menge der in Bezug auf A_R und d konstruierbaren Tupel $\{\mu \mid \mu : A_R \to d\}$ ist und

(iii) r über Universum d alle lokalen semantischen Bedingungen aus SC_R erfüllt.

Eine Relationeninstanz r über einem Universum d zu einem gegebenen Relationenschema $\langle R|A_R|SC_R\rangle$ ist also eine Menge von Tupeln. Dabei entspricht ein Tupel wiederum einer Funktion mit Urbild-Menge A_R, die jedem Attribut $a_j \in A_R$ des zugrunde liegenden Relationenschemas einen konkreten Wert aus der Menge d des Universums von r zuweist. Demnach gilt die Beziehung $\mu[a_j] = v$, wenn Attribut $a_j \in A_R$ in Tupel μ den Wert $v \in d$ zugewiesen bekommt.

Mensch	**Nachname**	**Vorname**	**Geschlecht**
	Bond	James	m
	Goldfinger	Auric	m
	King	Elektra	w
	Masterson	Jill	w

Abbildung 2.1: Beispielhafte Relationeninstanz *mensch*

Ein Beispiel für eine Relationeninstanz ist in Abbildung 2.1 gegeben. Diese Relationeninstanz *mensch* ist eine gültige Ausprägung zu einem Relationenschema

$$\langle Mensch | A_{\text{Mensch}} | SC_{\text{Mensch}} \rangle$$

mit Relationensymbol $Mensch$. Dabei enthält dieses Relationenschema die Attribute `Nachname`, `Vorname` und `Geschlecht` in der mit A_{Mensch} bezeichneten Menge der Attribute und legt damit die Struktur der Daten fest, die nach diesem Schema gespeichert werden können. Folglich müssen alle Tupel einer zu diesem Relationenschema gebildeten Relationeninstanz jedem dieser drei Attribute einen Wert zuweisen und haben damit eine geeignete Struktur, um (sehr grundlegende) Eigenschaften eines Menschen zu speichern.

Wie oben bereits erläutert, ist die in Abbildung 2.1 dargestellte Relationeninstanz *mensch* eine mögliche gültige Ausprägung zu dem betrachteten Relationenschema $\langle Mensch | A_{\text{Mensch}} | SC_{\text{Mensch}} \rangle$. In dieser Relationeninstanz sind vier Tupel enthalten und jedes dieser Tupel ist eine Funktion, die jedes der Attribute des zugrunde liegenden Relationenschemas auf einen Wert des Universums abbildet. So existiert in *mensch* beispielsweise ein Tupel μ, das

- das Attribut `Nachname` auf den Wert `Bond`,
- das Attribut `Vorname` auf den Wert `James` und
- das Attribut `Geschlecht` auf den Wert `m`

abbildet. Das bedeutet, dass unter anderem die Konstantenzeichen `Bond`, `James` und `m` in dem Universum, über dem *mensch* gebildet ist, enthalten sein müssen.

Unter der Annahme, dass der Vorname eines Menschen eindeutig das Geschlecht dieses Menschen bestimmt, kann zudem als Beispiel für eine lokale semantische Bedingung in SC_{Mensch} die funktionale Abhängigkeit

$$\{Vorname\} \rightarrow \{Geschlecht\}$$

definiert werden. Diese lässt es nicht zu, dass in einer Relationeninstanz, die eine gültige Ausprägung zu Relationenschema $\langle Mensch|A_{\text{Mensch}}|SC_{\text{Mensch}}\rangle$ ist, ein möglicher Vorname mit verschiedenen Geschlechtern in Relation steht. So kann in der oben vorgestellten Relationeninstanz *mensch* beispielsweise kein Tupel ν mit

- $\nu[\texttt{Vorname}] = \texttt{James}$ und
- $\nu[\texttt{Geschlecht}] = \texttt{w}$

existieren, solange in *mensch* das oben vorgestellte Tupel μ existiert.

2.1.2 Datenbankschema und Datenbankinstanz

Auf der Basis der Definitionen des Relationenschemas und der Relationeninstanz können im Folgenden die Konzepte des relationalen Datenbankschemas und der darauf aufbauenden relationalen Datenbankinstanz definiert werden.

Definition 2.3 (Relationales Datenbankschema) *Ein mit dem Namen RS bezeichnetes relationales Datenbankschema ist definiert durch*

$$RS = \langle\langle R_1|A_{R_1}|SC_{R_1}\rangle, \ldots, \langle R_n|A_{R_n}|SC_{R_n}\rangle \mid SC \mid \alpha\rangle \ ,$$

wobei $\langle R_i|A_{R_i}|SC_{R_i}\rangle$, $i \in \{1, \ldots, n\}$, jeweils ein Relationenschema nach Definition 2.1 ist, dessen Relationensymbol $R_i \in \mathcal{R}$ nicht als Relationensymbol eines anderen Relationenschemas dieses Datenbankschemas dient. Mit SC wird die Menge der globalen semantischen Bedingungen bezeichnet und die Funktion

$$\alpha : \bigcup_{i=1,\ldots,n} A_{R_i} \rightarrow \mathcal{B}$$

legt fest, welches Attribut welchem semantischen Bereichsnamen zugeordnet ist.

Ein relationales Datenbankschema besteht also in erster Linie aus einer Menge von Relationenschemata. Zwischen diesen Relationenschemata kann es Abhängigkeiten geben, die in der Menge SC der globalen semantischen Bedingungen des relationalen Datenbankschemas definiert werden. Ein Beispiel für globale semantische Bedingungen sind sogenannte Enthaltenseinsabhängigkeiten: Diese garantieren für Datenbankinstanzen (siehe unten) zu einem relationalen Datenbankschema RS, dass bestimmte Attributwerte einer Relationeninstanz zu einem bestimmten Relationenschema $\langle R_i|A_{R_i}|SC_{R_i}\rangle$ aus RS auch in einer Relationeninstanz zu einem bestimmten anderen Relationenschema $\langle R_j|A_{R_j}|SC_{R_j}\rangle$ aus RS enthalten sind.

Zudem werden alle in einem relationalen Datenbankschema enthaltenen Attribute über die Funktion α getypt, indem jedes Attribut einem semantischen Bereichsnamen zugeordnet wird. Da des Weiteren jeder semantische Bereichsname über die globale Funktion $\beta : \mathcal{B} \rightarrow \wp(\mathcal{K})$ einer Menge von Konstantenzeichen zugeordnet wird, ist über die Konkatenation $\beta \circ \alpha$ der Funktionen α und β für jedes Attribut eine zulässige Menge von Konstantenzeichen definiert.

Damit ist jetzt noch die Idee der zu einem Datenbankschema gehörenden relationalen Datenbankinstanz zu definieren, um die Grundlagen der relationalen Datenbanktheorie zu vervollständigen.

Definition 2.4 (Relationale Datenbankinstanz) *Sei gemäß Definition 2.3 ein relationales Datenbankschema*

$$RS = \langle \langle R_1 | A_{R_1} | SC_{R_1} \rangle, \ldots, \langle R_n | A_{R_n} | SC_{R_n} \rangle \mid SC \mid \alpha \rangle$$

gegeben. Eine relationale Datenbankinstanz $db = (r_1, r_2, \ldots, r_n)$ über dem Universum d ist genau dann eine gültige Ausprägung zu Schema RS, wenn

(i) d eine endliche Teilmenge der Konstantenzeichen $\mathcal{K}$ ist,

(ii) für $i \in \{1, \ldots, n\}$ jeweils r_i gemäß Definition 2.2 eine gültige Relationeninstanz über dem Universum d zu Relationenschema $\langle R_i | A_{R_i} | SC_{R_i} \rangle$ ist,

(iii) die Relationeninstanzen $r_1, r_2, \ldots, r_n$ über dem Universum d alle globalen semantischen Bedingungen aus SC erfüllen und

(iv) für alle $i \in \{1, \ldots, n\}$ stets gilt: $\forall \mu \in r_i : \forall a_j \in A_{R_i} : \mu[a_j] \in \beta \circ \alpha(a_j)$.

Damit ist eine relationale Datenbankinstanz eine konkrete Ausprägung zu einem Datenbankschema, die für jedes Relationenschema dieses Datenbankschemas eine Relationeninstanz enthält und garantiert, dass die durch das Datenbankschema vorgegebenen globalen semantischen Bedingungen eingehalten werden. Zusätzlich wird durch Bedingung (iv) aus Definition 2.4 sichergestellt, dass die durch das Datenbankschema vorgegebene Typisierung der Attribute eingehalten wird: Jedes in der Datenbankinstanz vorkommende Tupel μ darf einem Attribut a_j des Datenbankschemas ausschließlich Werte aus den Konstantenzeichen zuordnen, die gemäß der globalen Funktion β für den (im Datenbankschema durch α bestimmten) semantischen Bereichsnamen von a_j zulässig sind.

2.2 Funktionale Abhängigkeiten

In dem relationalen Datenmodell aus Kapitel 2.1 wird stets zwischen Schema und Instanz unterschieden: Während auf der Schema-Ebene die Form der Repräsentation von Information festgelegt wird, ist in Instanzen konkrete Information in der durch ein zugrunde liegendes Schema vorgegebenen Form enthalten. Dabei ist es für viele Datenbankanwendungen sinnvoll, in einem Relationenschema zusätzlich zu der Wahl einer geeigneten Menge von Attributen auch gewisse Restriktionen für Wertebelegungen dieser Attribute zu fordern, die in einer Relationeninstanz zu diesem Relationenschema eingehalten werden sollen (vgl. [1, Kap. 8.1]).

Dazu können in einem Relationenschema $\langle R|A_R|SC_R\rangle$ gemäß Definition 2.1 lokale semantische Bedingungen in der Menge SC_R angegeben werden. Zu diesen lokalen semantischen Bedingungen, die für ein Relationenschema definiert werden können, zählen unter anderem sogenannte funktionale Abhängigkeiten, die nach [1, Kap. 8.2] und [7, Kap. 15.1] folgendermaßen definiert sind:

Definition 2.5 (Funktionale Abhängigkeit) *Sei $\langle R|A_R|SC_R\rangle$ ein Relationenschema gemäß Definition 2.1 mit der Attributmenge A_R und einer Menge $\Sigma \subseteq SC_R$ von funktionalen Abhängigkeiten. Eine funktionale Abhängigkeit aus Σ hat die Form $A \to B$, wobei $A \subseteq A_R$ und $B \subseteq A_R$ gilt.*
In einer gemäß Definition 2.2 gebildeten Relationeninstanz r zu Relationenschema $\langle R|A_R|SC_R\rangle$ wird eine solche funktionale Abhängigkeit $A \to B$ genau dann erfüllt, wenn für alle Tupel $\mu, \nu \in r$, für die $\mu\lceil A = \nu\lceil A$ gilt, jeweils auch $\mu\lceil B = \nu\lceil B$ erfüllt wird. Dabei bezeichnet $\mu\lceil A$ die Einschränkung eines Tupels μ auf eine Attributmenge A (siehe [7, Kap. 8.1]).
Eine Menge $\Sigma \subseteq SC_R$ von funktionalen Abhängigkeiten wird genau dann in r erfüllt, wenn in r jede funktionale Abhängigkeit aus Σ erfüllt wird.

Damit werden in einer beliebigen Relationeninstanz r, die zu einem Relationenschema $\langle R|A_R|SC_R\rangle$ gebildet ist, die Werte der Attribute aus $B \subseteq A_R$ eindeutig durch die Werte der Attribute aus $A \subseteq A_R$ bestimmt, wenn für $\langle R|A_R|SC_R\rangle$ eine funktionale Abhängigkeit $A \to B$ in SC_R definiert ist [7, Kap. 15.1]. Es kann hier also auf der Ebene des Relationenschemas $\langle R|A_R|SC_R\rangle$ gefordert werden, dass eine Relationeninstanz r zu $\langle R|A_R|SC_R\rangle$, die ein Tupel μ enthält, *kein* weiteres Tupel ν enthalten darf, das zwar bezogen auf die Werte aller Attribute aus $A \subseteq A_R$ identisch zu μ ist, sich aber durch mindestens einen Wert für ein Attribut aus $B \subseteq A_R$ von μ unterscheidet. Ein einfaches Beispiel für eine funktionale Abhängigkeit wird bereits in Kapitel 2.1.1 als Beispiel für eine mögliche lokale semantische Bedingung in dem Relationenschema $\langle Mensch|A_{\text{Mensch}}|SC_{\text{Mensch}}\rangle$ gegeben.

3 Fragmentierte Speicherung von Datenbankinstanzen

Wie in Kapitel 1.1 erläutert wird, ist Information in unserer heutigen Informationsgesellschaft eine bedeutsame Ressource, die es zu schützen gilt. Zudem hegt man aber auch den Wunsch, diese Ressource möglichst kostengünstig durch externe Dienstleister verwalten zu lassen. Um diese offensichtlich konfliktbehafteten Anforderungen vereinen zu können, existiert das Konzept der fragmentierten Speicherung von Datenbankinstanzen. Dieses soll durch Aufteilen einer Datenbankinstanz in mehrere Fragment-Instanzen den nicht autorisierten Zugriff auf vertrauliche Information auch dann unterbinden, wenn die Fragment-Instanzen zu einer Datenbankinstanz (teilweise) auf nicht vertrauenswürdigen Servern gespeichert werden.

Zur Umsetzung dieses Konzepts existieren verschiedene konkrete Vorschläge, welche im Folgenden erläutert werden. Dazu wird einleitend in Kapitel 3.1 das allgemeine Konzept der (vertikalen) Fragmentierung von Relationenschemata vorgestellt, welches allen konkreten Ansätzen zu Grunde liegt. Anschließend werden in den Kapiteln 3.2, 3.3 und 3.4 dann drei existierende konkrete Verfahren, wie Zugriffsschutz durch Fragmentierung erzielt werden kann, beschrieben.

3.1 Vertraulichkeit durch Fragmentierung

Eine konventionelle Möglichkeit, Daten, die auf Servern externer Dienstleister gespeichert werden sollen, vor unbefugtem Zugriff zu schützen, besteht in der Verschlüsselung dieser Daten auf der Seite des Clients mit geeigneten kryptographischen Verfahren, bevor diese Daten auf den Server übertragen werden. Solche Verfahren werden beispielsweise in [22] oder [31] vorgeschlagen. Nachteilig an dieser Vorgehensweise ist jedoch, dass es bei geeignet starker Verschlüsselung oft nur bedingt möglich ist, Anfragen direkt auf Seite des Servers auf den verschlüsselten Daten auswerten zu lassen [2, Abschnitt 1]. Dies ist aber wünschenswert, um einen möglichst großen Teil des Rechenaufwands, der bei der Beantwortung von Anfragen anfällt, direkt von den verwendeten Servern bewältigen zu lassen. Bei diesen kann man üblicherweise davon ausgehen, dass ausreichend Rechenleistung zur Verfügung

steht, während auf der Seite des Clients auch schwächere (z.B. mobile, batteriebetriebene) Geräte zum Einsatz kommen können [24, Abschnitt 1].

Bei dem Einsatz kryptographischer Verfahren werden dabei stets die jeweiligen Daten selbst geschützt. Oft sind aber gerade in Datenbanken, in denen verschiedene Aspekte der zu verwaltenden Information anhand ihrer semantischen Bezüge zueinander strukturiert gespeichert werden, nur Assoziationen zwischen verschiedenen Daten vertraulich, während die einzelnen Daten für sich betrachtet nicht besonders schützenswert sind [27, Abschnitt 1]. Wenn man beispielsweise die Datenbankrelation eines Krankenhauses betrachtet, in der gespeichert wird, welche Patienten aufgrund welcher Krankheit behandelt wurden, dann ist die Menge der behandelten Krankheiten für sich genommen überhaupt nicht vertraulich und kann problemlos veröffentlicht werden. Ebenso ist die reine Information, welche Patienten in dem betrachteten Krankenhaus behandelt wurden, in der Regel auch nur bedingt sensibel. Hochgradig sensibel ist jedoch die Assoziation, welcher Patient aufgrund welcher Krankheit behandelt wurde.

Auf Basis dieser Betrachtungen kann man deshalb als Alternative zum ausschließlichen Einsatz von Kryptographie versuchen, vertrauliche Assoziationen zwischen Aspekten von strukturiert gespeicherter Information zu verbergen, indem man die Informations-Aspekte, zwischen denen eine Assoziation zu schützen ist, getrennt von einander aufbewahrt. Im Kontext einer Datenbankrelation bedeutet dies, dass bestimmte Attribute (bzw. Spalten) dieser Relation getrennt von einander gespeichert werden müssen. Dazu wird eine Datenbankrelation gemäß bestimmter Vertraulichkeitsanforderungen (siehe Kapitel 3.1.2) in mehrere Teile aufgeteilt.

3.1.1 Fragmentierung von Datenbankrelationen

Es gibt mit der sogenannten horizontalen und der sogenannten vertikalen Fragmentierung grundsätzlich zwei Möglichkeiten, eine Datenbankrelation zu fragmentieren. Bei der horizontalen Fragmentierung werden die Tupel einer konkreten Relationeninstanz anhand bestimmter Selektions-Kriterien auf mehrere (neu entstehende) Relationeninstanzen aufgeteilt, die auch Fragment-Instanzen genannt werden [38, Kap. 5.3.1]. Die in den Schemata der Fragment-Instanzen enthaltene Menge an Attributen ist dabei jeweils identisch zu der Attributmenge des Schemas der ursprünglichen Relationeninstanz. Jede der entstandenen Fragment-Instanzen enthält demnach eine Teilmenge der Tupel der fragmentierten Relationeninstanz. Bezogen auf die tabellarische Darstellung einer Relationeninstanz wird diese also horizontal mit Hilfe von Selektionen in Fragment-Instanzen zerlegt, so dass die Vereinigung der entstandenen Fragment-Instanzen wieder der ursprünglichen Relationeninstanz entspricht.

Bei der vertikalen Fragmentierung werden hingegen die Attribute des Schemas einer zu fragmentierenden Relationeninstanz auf mehrere (neu entstehende) Relationenschemata aufgeteilt, die auch Fragmente genannt werden [38, Kap. 5.3.2]. Zu jedem der entstandenen Fragmente, von denen jedes eine Teilmenge der Attribute des ursprünglichen Relationenschemas enthält, wird eine konkrete Fragment-Instanz gebildet. Diese lässt sich mit Hilfe der Projektion der ursprünglichen Relationeninstanz auf die Attribute des jeweils betrachteten Fragments konstruieren. Wenn die Fragmentierung beispielsweise derart erfolgt, dass jedes der Fragmente die Attribute eines Primärschlüssels des ursprünglichen Relationenschemas enthält, kann die ursprüngliche Relationeninstanz mit Hilfe des natürlichen Verbundes wieder aus den entstandenen Fragment-Instanzen rekonstruiert werden.

Im Rahmen dieser Ausarbeitung soll ausschließlich die vertikale Fragmentierung einzelner Datenbankrelationen betrachtet werden. Aus diesem Grund ist stets die vertikale Fragmentierung gemeint, wenn nachfolgend von einer nicht näher spezifizierten Fragmentierung gesprochen wird. Diese vertikale Fragmentierung eines Relationenschemas[1] $\langle R|A_R|\ \ \rangle$ und einer Relationeninstanz r zu diesem Schema soll zuerst genauer formalisiert werden [27, Abschnitt 3].

Definition 3.1 (Vertikale Fragmentierung) *Sei $\langle R|A_R|\ \ \rangle$ ein gemäß Definition 2.1 aufgebautes Relationenschema mit Relationensymbol R und der endlichen Menge A_R an Attributen. Eine vertikale Fragmentierung $\mathcal{F}$ dieses Relationenschemas ist durch die Menge*

$$\mathcal{F} = \{\langle F_1|A_{F_1}|SC_{F_1}\rangle, \langle F_2|A_{F_2}|SC_{F_2}\rangle, \ldots, \langle F_n|A_{F_n}|SC_{F_n}\rangle\}$$

gegeben, wobei $\langle F_i|A_{F_i}|SC_{F_i}\rangle$ für jedes $i \in \{1, \ldots, n\}$ jeweils ein Relationenschema gemäß Definition 2.1 ist, in dem $A_{F_i} \subseteq A_R$ gilt. Ein solches Relationenschema $\langle F_i|A_{F_i}|SC_{F_i}\rangle$ wird auch Fragment der Fragmentierung $\mathcal{F}$ genannt.
Wenn r gemäß Definition 2.2 eine Relationeninstanz zu Schema $\langle R|A_R|\ \ \rangle$ ist, kann eine Fragment-Instanz f_i zu Fragment $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ bezüglich r durch die Projektion $f_i = \{\mu\lceil A_{F_i} \mid \mu \in r\}$ gebildet werden. Dabei bezeichnet $\mu\lceil A$ die Einschränkung eines Tupels μ auf eine Attributmenge A (siehe [7, Kap. 8.1]).

Dies ist die (bezogen auf den Kontext dieser Ausarbeitung) allgemeinste Definition der vertikalen Fragmentierung, die sich lediglich auf die Aufteilung eines Relationenschemas und einer Relationeninstanz beschränkt. Deshalb ist diese allgemeinste Definition vom Ansatz her zu allen nachfolgend vorgestellten konkreten Verfahren der Fragmentierung einer Datenbankrelation im Sinne des Vertraulichkeitsschutzes

[1] Da mögliche lokale semantische Bedingungen eines Relationenschemas an dieser Stelle nicht beachtet werden, wird die entsprechende dritte Stelle im Tripel leer gelassen.

kompatibel. Für jedes der vorgestellten konkreten Verfahren wird diese Definition dann an entsprechender Stelle geeignet spezialisiert werden.

Beim Fragmentieren einer Datenbankrelation ist es im Allgemeinen wünschenswert, einige weitere Eigenschaften zu fordern, um einerseits die Konsistenz des fragmentierten Datenbestandes zu gewährleisten und andererseits die Größe der erzeugten Fragmente gering zu halten [38, Kap. 5.2.4]. Um diese Ziele zu erreichen, werden für die Fragmentierung[2] $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ eines Relationenschemas R in der Regel die folgenden drei Eigenschaften gefordert:

Vollständigkeit: Jedes Attribut des ursprünglichen Relationenschemas R muss in mindestens einem Fragment $F_i \in \mathcal{F}$ enthalten sein, damit bei der Fragmentierung keine Attribute von R verloren gehen. Dadurch ist gewährleistet, dass jeder Attributwert aus einer ursprünglichen Relationeninstanz r zu Schema R auch in mindestens einer Fragment-Instanz f_i enthalten ist, die zu einem Fragment $F_i \in \mathcal{F}$ bezüglich r gebildet ist. Dabei können die Werte bestimmter Attribute in einzelnen Fragment-Instanzen auch in verschlüsselter Form repräsentiert werden. Da eine Fragment-Instanz f_i zu Fragment $F_i \in \mathcal{F}$ bezüglich r einer Projektion von r auf die Attributmenge von F_i entspricht, bleiben innerhalb einer Fragment-Instanz f_i auch – abgesehen von eventuell verschlüsselt repräsentierten Werten – alle Wertekombinationen aus Tupeln von r hinsichtlich der Attributmenge von F_i korrekt erhalten.

Rekonstruierbarkeit: Die Fragmentierung $\mathcal{F}$ soll so beschaffen sein, dass die zu einer Relationeninstanz r gemäß $\mathcal{F}$ gebildeten Fragment-Instanzen $f_1, f_2, \ldots, f_n$ wieder derart korrekt zusammengesetzt werden können, dass für jedes Tupel in der Kombination von $f_1, f_2, \ldots, f_n$ genau ein entsprechendes Tupel in der ursprünglichen Relationeninstanz r, die fragmentiert wurde, existiert. Diese Eigenschaft garantiert, dass bei der Fragmentierung die semantischen Bezüge zwischen den Daten einzelner Fragmente (und der jeweils in diesen Daten repräsentierten Information) nicht verloren gehen. Die Forderung nach Rekonstruierbarkeit ist beispielsweise erfüllt, wenn jedes Fragment $F_i \in \mathcal{F}$ die Attribute eines Primärschlüssels aus R enthält, so dass die ursprüngliche Instanz r über den natürlichen Verbund $f_1 \bowtie f_2 \bowtie \ldots \bowtie f_n$ wieder hergestellt werden kann [38, Kap. 5.3.2].

Überlappungsminimalität: Jedes Attribut des ursprünglichen Relationenschemas R soll möglichst in maximal einem Fragment $F_i \in \mathcal{F}$ enthalten sein, sofern es aufgrund der Forderung nach Rekonstruierbarkeit nicht notwendig ist, dass

[2] Wenn sich aus dem Kontext heraus klar ergibt, dass ein Fragment $\langle F_i | A_{F_i} | SC_{F_i} \rangle$ gemeint ist, wird dieses Fragment im Folgenden oft abkürzend durch sein Relationensymbol F_i bezeichnet. Analog dazu wird ein Relationenschema $\langle R | A_R | SC_R \rangle$ oft abkürzend durch R bezeichnet.

Patient	**SSN**	**Name**	**Geburtstag**	**Plz**	**Krankheit**	**Arzt**
	12345	Hellmann	03.01.1981	94142	Bluthochdruck	White
	98765	Dooley	07.10.1953	94141	Fettleibigkeit	Warren
	24689	McKinley	12.02.1952	94139	Bluthochdruck	White
	13579	Ripley	03.01.1981	94139	Fettleibigkeit	Warren

Abbildung 3.1: Instanz *patient* zu Relationenschema *Patient*

dieses Attribut in mehr als einem Fragment enthalten ist. Durch die Eigenschaft der Überlappungsminimalität kommt der grundsätzliche Gedanke der Fragmentierung – in Form des Aufteilens eines Relationenschemas in mehrere Teile – zum Ausdruck. Wenn kein Attribut aus R in mehr als einem Fragment enthalten ist, ist größtmögliche Überlappungsminimalität gegeben.

In Abbildung 3.1 ist beispielhaft eine Relationeninstanz *patient* zu dem Relationenschema $\langle Patient | A_{\text{Patient}} | SC_{\text{Patient}} \rangle$ gegeben [27]. Die konkrete Menge A_{Patient} der Attribute dieses Relationenschemas kann dabei aus der Abbildung entnommen werden. In dieser ist auch zu sehen, dass das Attribut `SSN` in SC_{Patient} als Primärschlüssel vereinbart ist. Eine mögliche Fragmentierung des Relationenschemas *Patient* ist durch $\mathcal{F} = \{F_1, F_2, F_3\}$ gegeben, wobei $\{\texttt{Name}\}$ die Attributmenge von F_1, $\{\texttt{Geburtstag}, \texttt{Plz}\}$ die Attributmenge von F_2 und $\{\texttt{Krankheit}, \texttt{Arzt}\}$ die Attributmenge von F_3 ist. Jedes Fragment enthält also wie gefordert eine Teilmenge der Attribute des fragmentierten Relationenschemas *Patient*. Bezogen auf die Relationeninstanz *patient* zu Relationenschema *Patient* ergeben sich bei der hier gewählten Fragmentierung $\mathcal{F}$ die in Abbildung 3.2 dargestellten Fragment-Instanzen f_1, f_2 und f_3 zu den entsprechenden Fragmenten F_1, F_2 und F_3.

Von den oben beschriebenen Eigenschaften der Vollständigkeit, der Rekonstruierbarkeit und der Überlappungsminimalität ist für die Fragmentierung aus Abbildung 3.2 ausschließlich die größtmögliche Überlappungsminimalität erfüllt, da kein Attribut des Relationenschemas *Patient* in mehr als einem Fragment enthalten ist. Die Eigenschaft der Vollständigkeit ist nicht erfüllt, da das Attribut `SSN` aus *Patient* in keinem Fragment vorzufinden ist. Ebenso ist die Rekonstruierbarkeit nicht gegeben, weil der Primärschlüssel `SSN`, über den eine eindeutige Identifikation der jeweils zusammengehörenden Teile eines Tupels aus *patient* möglich wäre, in keinem Fragment enthalten ist. Wäre dieser Schlüssel in jedem Fragment enthalten, könnte die ursprüngliche Instanz *patient* über den natürlichen Verbund $f_1 \bowtie f_2 \bowtie f_3$ rekonstruiert werden.

Aufgrund des fehlenden Primärschlüssels `SSN` in F_3 sind auch in der zugehörigen

F_1	**Name**
	Hellmann
	Dooley
	McKinley
	Ripley

F_2	**Geburtstag**	**Plz**
	03.01.1981	94142
	07.10.1953	94141
	12.02.1952	94139
	03.01.1981	94139

F_3	**Krankheit**	**Arzt**
	Bluthochdruck	White
	Fettleibigkeit	Warren

Abbildung 3.2: Mögliche Fragment-Instanzen nach Fragmentierung von *patient*

Fragment-Instanz f_3 nur zwei Tupel vorhanden, obwohl die ursprüngliche Relationeninstanz *patient* vier Tupel enthält. Da sich das erste und das dritte Tupel aus *patient* bezüglich der Attribute `Krankheit` und `Arzt` nicht unterscheiden, werden diese zwei Tupel in f_3 durch die Projektion von *patient* auf die Attributmenge $\{\texttt{Krankheit}, \texttt{Arzt}\}$ ununterscheidbar. Da f_3 als Relationeninstanz nicht zwei identische Tupel enthalten kann, werden diese beiden Tupel zu einem vereint. Für das zweite und das vierte Tupel aus *patient* wird analog dazu verfahren.

3.1.2 Definition von Vertraulichkeitsanforderungen

In das oben vorgestellte Konzept der Fragmentierung von Datenbankrelationen wurde mit der Motivation eingeführt, dass durch eine geeignete Fragmentierung einer Datenbankrelation Vertraulichkeitsanforderungen hinsichtlich vertraulicher Assoziationen erfüllt werden können. Um dieses Ziel zu erreichen, bedarf es einer Möglichkeit, die gewünschten Vertraulichkeitsanforderungen formal definieren zu können. Diese Möglichkeit bietet das Konzept der sogenannten Vertraulichkeits-Constraints, das im Folgenden vorgestellt wird [27, Abschnitt 2].

Definition 3.2 (Vertraulichkeits-Constraints) *Sei $\langle R|A_R|\ \rangle$ ein Relationenschema gemäß Definition 2.1 mit der endlichen Menge A_R an Attributen. Ein Vertraulichkeits-Constraint zu $\langle R|A_R|\ \rangle$ ist eine Teilmenge $c \subseteq A_R$ der Attributmenge dieses Relationenschemas. Dabei heißt ein Vertraulichkeits-Constraint c einfach, wenn es einer einelementigen Teilmenge entspricht, und assoziierend, wenn es einer Teilmenge c mit Kardinalität $|c| > 1$ entspricht.*
Eine Menge $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ von Vertraulichkeits-Constraints heißt genau dann wohldefiniert, wenn für alle $c_i, c_j \in \mathcal{C}$ mit $i \neq j$ jeweils $c_i \not\subset c_j$ gilt.

Bezogen auf den semantischen Kontext der Fragmentierung eines Relationenschemas ist ein Vertraulichkeits-Constraint $\{a_1, \ldots, a_k\} \in \mathcal{C}$ eine Teilmenge von Attributen dieses Relationenschemas, die (genauer gesagt, deren über Tupel zugeord-

nete Werte) in dieser Kombination für einen nicht autorisierten Betrachter nicht zusammen sichtbar sein dürfen. Dabei muss zwischen einfachen und assoziierenden Vertraulichkeits-Constraints unterschieden werden.

Im Falle eines assoziierenden Vertraulichkeits-Constraints bedeutet dies, dass diese Menge $\{a_1, \ldots, a_k\}$ an Attributen nicht (sichtbar) gemeinsam in einem Fragment, dessen zugehörige Fragment-Instanz für einen nicht autorisierten Betrachter lesbar ist, gespeichert werden dürfen. Es muss immer eine mindestens einelementige, echte Teilmenge $\tilde{a} \subset \{a_1, \ldots, a_k\}$ so gespeichert werden, dass eine Verknüpfung der diesen Attributen $\tilde{a}$ zugeordneten Werte mit den Werten der restlichen Attribute $\{a_1, \ldots, a_k\} \setminus \tilde{a}$ für einen solchen Betrachter nicht möglich ist. Dieser darf demnach zur Durchsetzung des Vertraulichkeits-Constraints $\{a_1, \ldots, a_k\} \in \mathcal{C}$ nicht die Möglichkeit haben, die Projektion der ursprünglichen Relationeninstanz auf die Attributmenge $\{a_1, \ldots, a_k\}$ korrekt rekonstruieren zu können.

Als Folge dessen darf für einen nicht autorisierten Betrachter damit die in Kapitel 3.1.1 diskutierte Eigenschaft der Rekonstruierbarkeit der ursprünglichen Relationeninstanz auf Basis der für ihn sichtbaren Fragment-Instanzen nicht gelten, um die (gemäß der Vertraulichkeits-Constraints) zu schützenden Assoziationen nicht offen zu legen. Dazu können ihm komplette Fragment-Instanzen vorenthalten werden oder seine Sichtweise auf bestimmte Fragment-Instanzen so eingeschränkt werden (z.B. durch Verschlüsselung der Werte einzelner Attribute), dass eine Kombination von je zwei Fragment-Instanzen, die eine vertrauliche Assoziation offen legen würde, für ihn nicht korrekt im Sinne der ursprünglichen Relationeninstanz möglich ist.

Im Falle eines einfachen Vertraulichkeits-Constraints $\{a\} \in \mathcal{C}$ ist nicht eine Assoziation zwischen Werten von Attributen zu schützen, sondern die dem Attribut a zugeordneten Werte selbst. Zu diesem Zweck müssen die a zugeordneten Werte in einer dazu geeigneten Fragment-Instanz so gespeichert werden, dass sie für einen nicht autorisierten Betrachter nicht einsehbar sind. Das heißt, dass aus Sicht dieses Betrachters zum Schutz von Attributwerten, die für ihn aufgrund von einfachen Vertraulichkeits-Constraints nicht sichtbar sein dürfen, die in Kapitel 3.1.1 diskutierte Eigenschaft der Vollständigkeit der Fragmentierung nicht gelten darf.

Für einen autorisierten Benutzer müssen hingegen stets die Eigenschaften der Vollständigkeit und der Rekonstruierbarkeit der Fragmentierung gewahrt bleiben, damit dieser bei Bedarf Zugriff auf die komplette ursprüngliche Relationeninstanz bekommen kann. Dazu hat er im Gegensatz zu einem nicht autorisierten Betrachter je nach Konzept beispielsweise Zugriff auf die komplette Menge an Fragment-Instanzen oder ist im Besitz kryptographischer Schlüssel, um verschlüsselt gespeicherte Attributwerte wieder korrekt entschlüsseln zu können.

$$
\begin{aligned}
c_0 &= \{\texttt{SSN}\} \\
c_1 &= \{\texttt{Name}, \texttt{Geburtstag}\} \\
c_2 &= \{\texttt{Name}, \texttt{Plz}\} \\
c_3 &= \{\texttt{Name}, \texttt{Krankheit}\} \\
c_4 &= \{\texttt{Name}, \texttt{Arzt}\} \\
c_5 &= \{\texttt{Geburtstag}, \texttt{Plz}, \texttt{Krankheit}\} \\
c_6 &= \{\texttt{Geburtstag}, \texttt{Plz}, \texttt{Arzt}\}
\end{aligned}
$$

Abbildung 3.3: Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für *Patient*

Die genaue Semantik eines Vertraulichkeits-Constraints kann an dieser Stelle noch nicht erläutert werden, da diese abhängig von dem gewählten konkreten Ansatz zum Erzielen des Vertraulichkeitsschutzes durch Fragmentierung ist. Aus diesem Grund wird die genaue Semantik eines Vertraulichkeits-Constraints in jedem der nachfolgenden Kapitel 3.2, 3.3 und 3.4, in denen diese konkreten Ansätze vorgestellt werden, jeweils bezogen auf den entsprechenden Ansatz definiert.

Im Rahmen der Definition von Vertraulichkeits-Constraints wird auch auf die Eigenschaft der Wohldefiniertheit von Constraint-Mengen eingegangen. Eine Menge $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ von Vertraulichkeits-Constraints heißt genau dann wohldefiniert, wenn kein Constraint $c_i \in \mathcal{C}$ einer Teilmenge der Attribute eines anderen Constraints $c_j \in \mathcal{C}$ entspricht. Angenommen, es gelte $c_i \subset c_j$. Wenn unter dieser Annahme eine Fragmentierung Constraint c_i beachtet, indem die Werte einer mindestens einelementigen Attributmenge $\tilde{a} \subset c_i$ (aus Sicht eines nicht autorisierten Betrachters) getrennt von den Werten der Attribute $c_i \setminus \tilde{a}$ gespeichert werden, beachtet diese Fragmentierung automatisch auch Constraint c_j. Dies ist der Fall, weil wegen $\tilde{a} \subset c_i$ und $c_i \subset c_j$ aufgrund der Transitivität der Teilmengen-Relation auch $\tilde{a} \subset c_j$ gilt und mit den zu $\tilde{a}$ gehörenden Attributwerten auch die Werte einer mindestens einelementigen Attributmenge $\tilde{a} \subset c_j$ getrennt von den Werten der Attribute $(c_j \cap c_i) \setminus \tilde{a}$ aus Constraint c_j gespeichert würden. Damit gilt unter der Annahme $c_i \subset c_j$, dass eine Fragmentierung, die c_i beachtet, grundsätzlich immer auch automatisch c_j beachtet und c_j braucht nicht gesondert in der Menge der Vertraulichkeits-Constraints aufgeführt zu werden.

Ein Beispiel für eine wohldefinierte Menge von Vertraulichkeits-Constraints für das Relationenschema *Patient* (siehe zugehörige Instanz in Abbildung 3.1) ist in Abbildung 3.3 gegeben [27, Abschnitt 2]. In dieser Constraint-Menge $\mathcal{C}$ ist mit c_0 ein einfaches Vertraulichkeits-Constraint enthalten, welches ausdrückt, dass alle Werte des Attributs `SSN` zu schützen sind. Die restlichen Constraints sind assoziierend und schützen damit Assoziationen zwischen Werten von Attributen. Dabei drücken die Constraints $c_1, \ldots, c_4$ aus, dass eine Assoziation zwischen dem Attribut `Name` und

einem beliebigen anderen Attribut sensibel ist. Ein Constraint $\{\texttt{Name}, \texttt{SSN}\}$ existiert dazu aber nicht explizit, weil diese Assoziation implizit schon durch das Constraint $c_0 = \{\texttt{SSN}\} \subset \{\texttt{Name}, \texttt{SSN}\}$ unterbunden wird und aufgrund der Wohldefiniertheit der Menge $\mathcal{C}$ in dieser nicht noch zusätzlich enthalten sein darf.

Zusätzlich zu den direkten Assoziationen zwischen dem Attribut `Name` und einem beliebigen anderen Attribut des Relationenschemas kann im oben vorgestellten Beispiel auch die Gefahr bestehen, dass eine Kombination mehrerer Attribute als sogenannter Quasi-Identifikator (siehe [37]) genutzt wird, um den Namen einer bestimmten Person zu erschließen und auf diese Weise sensible Assoziationen herstellen zu können. Selbst wenn eine eindeutige Identifikation einer einzelnen Person nicht möglich ist, besteht immer noch die Gefahr, dass die Gruppe der anhand des Quasi-Identifikators in Frage kommenden Personen stark eingeschränkt wird. Um diese Art von Schlussfolgerungen zu verhindern, sind in der Constraint-Menge $\mathcal{C}$ aus Abbildung 3.3 die Vertraulichkeits-Constraints c_5 und c_6 enthalten. Diese beugen der Möglichkeit vor, dass die Kombination der Werte aus den Attributen `Geburtstag` und `Plz` eines bestimmten Tupels als Quasi-Identifikator zum Erschließen des Namens eines Patienten genutzt wird, um diesen Patienten mit einem Wert der Attribute `Krankheit` oder `Arzt` in Verbindung bringen zu können.

3.1.3 Anfragen an fragmentierte Datenbankinstanzen

Nachdem in die grundlegenden Ideen, wie Vertraulichkeit durch Fragmentierung erzielt werden kann, eingeführt wurde, soll im Folgenden in Anlehnung an die einleitende Idee aus [2, Abschnitt 4.2] eine (triviale und nicht effiziente) Möglichkeit vorgestellt werden, wie Anfragen eines autorisierten Benutzers ausschließlich mit Hilfe der Fragment-Instanzen einer fragmentierten Relationeninstanz beantwortet werden können. Dabei wird davon ausgegangen, dass eine ursprüngliche Relationeninstanz r gemäß einer berechneten Fragmentierung $\mathcal{F}$ in die Fragment-Instanzen $f_1, f_2, \ldots, f_n$ zerlegt worden ist. Für diese Fragmentierung seien die Eigenschaften der Vollständigkeit und der Rekonstruierbarkeit erfüllt.

Weiter wird im Folgenden davon ausgegangen, dass eine Teilmenge $f_{i_1}, \ldots, f_{i_k}$ dieser Fragment-Instanzen auf Servern externer Dienstleister gespeichert wird, auf die über das Internet zugegriffen werden kann. Die restlichen Fragment-Instanzen $f_{i_{k+1}}, \ldots, f_{i_n}$ seien lokal (und damit sicher vor unautorisierten Zugriffen) gespeichert. Die verwendeten externen Server garantieren dabei stets die Integrität der gespeicherten Daten, gelten aber nicht als vertrauenswürdig im Hinblick auf das zu erreichende Schutzziel der Vertraulichkeit. Aufgrund dessen soll die gewünschte Vertraulichkeit durch Ansätze zur Fragmentierung erzielt werden [2, Abschnitt 2].

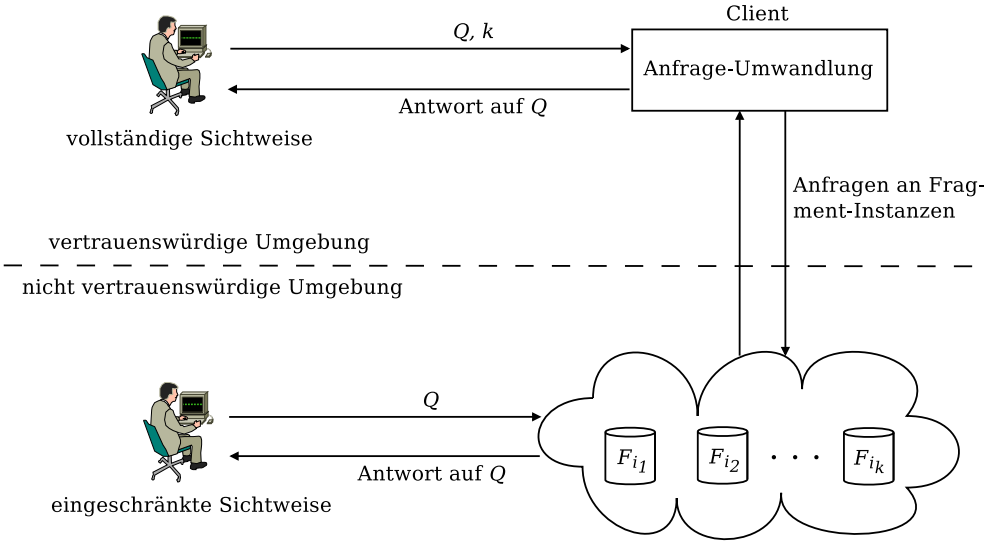

Abbildung 3.4: Ablauf von Anfragen an eine fragmentierte Relationeninstanz

Die grundsätzlichen Ideen, wie Anfragen an eine fragmentierte Relationeninstanz gestellt werden können, sind in Abbildung 3.4 dargestellt (vgl. [27, Abschnitt 8]). Ein autorisierter Benutzer, der eine vollständige Sichtweise auf die ursprüngliche Relationeninstanz haben soll, stellt eine Anfrage Q dabei immer an einen dafür vorgesehenen Client. Diese Anfrage kann der Benutzer so stellen, als ob er direkt eine Anfrage an die ursprüngliche Relationeninstanz r stellen würde. Er braucht beim Formulieren seiner Anfrage also die Fragmentierung der Relationeninstanz nicht in Betracht zu ziehen. Diese Aufgabe wird vom Client übernommen.

Unter der Annahme, dass die ursprüngliche Relationeninstanz r wieder korrekt über den natürlichen Verbund $r = f_1 \bowtie f_2 \bowtie \ldots \bowtie f_n$ rekonstruiert werden kann, kann der verwendete Client die Anfrage Q derart umformulieren, dass diese nicht als Anfrage an r, sondern als Anfrage an den natürlichen Verbund der Fragment-Instanzen, in die r zerlegt worden ist, interpretiert wird. Dazu soll hier so vorgegangen werden, dass im Vorfeld der Beantwortung der Anfrage Q die Fragment-Instanzen $f_{i_1}, \ldots, f_{i_k}$, die nach Voraussetzung extern verwaltet werden, komplett durch geeignete Anfragen von den verwendeten externen Servern heruntergeladen werden. Da die restlichen Fragment-Instanzen $f_{i_{k+1}}, \ldots, f_{i_n}$ nach Voraussetzung lokal verwaltet werden, stehen dem verwendeten Client damit alle Fragment-Instanzen, die zur korrekten Rekonstruktion der ursprünglichen Relationeninstanz r über den natürlichen Verbund benötigt werden, zur Verfügung.

Im Rahmen der Berechnung des natürlichen Verbundes $r = f_1 \bowtie f_2 \bowtie \ldots \bowtie f_n$ durch den Client muss beachtet werden, dass in einzelnen Fragment-Instanzen unter Umständen auch die Werte bestimmter Attribute verschlüsselt gespeichert werden. Dieser Fall wird im Rahmen dieser Diplomarbeit in zwei verschiedenen Ausprägungen betrachtet: In Kapitel 3.3 wird davon ausgegangen, dass alle verschlüsselt gespeicherten Attributwerte über ein symmetrisches Kryptographie-Verfahren mit demselben kryptographischen Schlüssel k verschlüsselt sind, der einem autorisierten Benutzer bekannt ist. Als Folge dessen können die verschlüsselt gespeicherten Attributwerte im Vorfeld der Berechnung des natürlichen Verbundes mit Hilfe des Schlüssels k, der durch den Benutzer zusammen mit der Anfrage Q an den Client übermittelt wird, durch den Client entschlüsselt werden.

In Kapitel 3.2 wird davon ausgegangen, dass jeder in einer Fragment-Instanz f verschlüsselt gespeicherte Attributwert v eines Attributs a, dessen Werte verschlüsselt zu speichern sind, jeweils mit einem individuellen kryptographischen Schlüssel k_v über ein symmetrisches Kryptographie-Verfahren verschlüsselt ist. Dabei wird der Schlüssel k_v in einer anderen – im Folgenden mit f' bezeichneten – Fragment-Instanz als der Wert v verwaltet. Zur Verwaltung der Schlüssel, mit denen die Attributwerte von a in f verschlüsselt sind, existiert in dem Fragment, zu dem die Fragment-Instanz f' gebildet ist, ein eigenes Attribut zur Verwaltung der Schlüssel für die Attributwerte von a. Im Vorfeld der Konstruktion des natürlichen Verbundes müssen durch den Client deshalb alle in f verschlüsselt gespeicherten Attributwerte von a mit dem jeweils entsprechenden Schlüssel aus f' entschlüsselt werden und die kryptographischen Schlüssel für die Attributwerte von a müssen aus der Fragment-Instanz f' mit Hilfe einer dazu geeigneten Projektion von f' entfernt werden.

Wie bereits einleitend erwähnt wurde, ist die oben vorgestellte Methode, Anfragen eines autorisierten Benutzers zu beantworten, denkbar ineffizient. Deshalb ist diese nur als Begründung dafür zu sehen, dass es prinzipiell möglich ist, Anfragen allein durch die Fragment-Instanzen einer fragmentierten Relationeninstanz beantworten zu können. Um Anfragen effizient beantworten zu können, ist es notwendig, im Rahmen einer Anfrageauswertung bereits möglichst viele Selektionen und Projektionen direkt auf den Servern, die die extern gespeicherten Fragment-Instanzen verwalten, durchzuführen (vgl. [2, Abschnitt 4.2]). Dadurch wird möglichst viel Rechenaufwand durch die verwendeten Server bewältigt und die zu übertragenden Datenmengen an den Client werden klein gehalten. Für jeden der nachfolgend vorgestellten konkreten Fragmentierungs-Ansätze werden dazu in der jeweils angegebenen Literatur Vorschläge unterbreitet, wie eine effiziente Anfrageauswertung für den jeweils betrachteten Ansatz gestaltet werden kann.

In dem Anfrage-Modell aus Abbildung 3.4 wird vorausgesetzt, dass der Client, der sowohl Zugriff auf alle vertraulichen Assoziationen als auch auf alle vertraulichen

Attributwerte hat, vollkommen sicher und vertrauenswürdig ist [2, Abschnitt 2]. Dies ist im Sinne der Verfügbarkeit notwendig, um einem Benutzer Anfragen zu ermöglichen, für die mehr als eine Fragment-Instanz benötigt werden oder für die auf verschlüsselte Werte zugegriffen werden muss, ohne dabei das Sicherheitskonzept zu gefährden. Das Problem, dass es bei einzelnen Komponenten eines Gesamtsystems oft notwendig ist, sich auf deren Vertrauenswürdigkeit zu verlassen, wird in der Literatur auch als „Root of Trust"-Problem bezeichnet [9, Kap. 8.2.7].

Nicht autorisierte Betrachter, die weder Zugriff auf den vertrauenswürdigen Client haben noch im Besitz eventuell vorhandener kryptographischer Schlüssel sind, haben, wie in Abbildung 3.4 zu sehen ist, lediglich eine eingeschränkte Sichtweise auf den Datenbestand, indem sie unter Umständen direkt Anfragen an einzelne Fragment-Instanzen stellen können [27, Abschnitt 8]. Die für nicht autorisierte Betrachter lesbaren Fragment-Instanzen sind aber gemäß der Vertraulichkeitsanforderungen so beschaffen, dass durch Zugriff auf diese die Vertraulichkeit zu schützender Assoziationen und Attributwerte nicht gefährdet werden kann.

Unter Umständen können gezielte Zugriffe eines nicht autorisierten Betrachters auf einzelne Fragment-Instanzen aber durchaus sinnvoll sein. In dem in Abbildung 3.2 gegebenen Beispiel einer Fragmentierung der Datenbankrelation *patient* könnte es zum Beispiel aus wissenschaftlichen Gesichtspunkten von Interesse sein, die Assoziationen zwischen Werten der Attribute `Geburtstag` und `Krankheit` auch nicht autorisierten Betrachtern zugänglich zu machen. Dadurch könnten Zusammenhänge zwischen dem Alter und der Krankheit eines Patienten erforscht werden, ohne dessen Identität preisgeben zu müssen. Es kann also sinnvoll sein, Assoziationen dieser Art bei Berechnung einer Fragmentierung gezielt aufrecht zu erhalten, sofern durch diese keine Vertraulichkeits-Constraints verletzt werden. Um diese Art von Verfügbarkeitsanforderungen bei Berechnung einer Fragmentierung berücksichtigen zu können, existiert die Idee, sogenannte Affinitäts-Werte für die Beziehung zwischen je zwei Attributen eines Relationenschemas zu definieren [27, Abschnitt 6].

3.2 Fragmentierung und nicht-kooperierende Partner

Nachdem in die grundlegenden Ideen, wie Vertraulichkeit in relationalen Datenbanken durch Fragmentierung von Datenbankrelationen erreicht werden kann, eingeführt wurde, sollen nun drei existierende konkrete Ansätze, die auf diesen Ideen aufbauen, vorgestellt werden. Eine Übersicht über diese Ansätze ist auch in [34] zu finden: Dort werden – neben weiteren, auf Kryptographie basierenden Verfahren zur sicheren Verwaltung von Daten auf Servern externer Dienstleister – die grundlegenden Ideen, auf denen jeder dieser drei Ansätze jeweils basiert, vorgestellt.

Der in diesem Kapitel beschriebene erste Ansatz wird in [2] vorgestellt und basiert auf der Annahme, dass zur Speicherung einer Datenbankrelation zwei von einander komplett unabhängige Server existieren, die in keiner Weise miteinander kooperieren. Diese Annahme beinhaltet auch, dass jeder nicht autorisierte Betrachter auf maximal einen der beiden Server Zugriff bekommen kann. Der Client hat in diesem Modell lediglich die Funktion, Anfragen eines autorisierten Benutzers zu beantworten. Insbesondere können keine Teile des zu verwaltenden Datenbestandes auf diesem persistent gespeichert werden. Das heißt, dass alle Fragment-Instanzen einer Fragmentierung auf den beiden externen, nicht vertrauenswürdigen Servern gespeichert werden müssen.

3.2.1 Fragmentierung der Datenbankrelation

Um eine Fragmentierung einer Datenbankrelation berechnen zu können, die unter der Voraussetzung zweier unabhängiger, nicht kooperierender Server Vertraulichkeit erzielt, muss erst einmal die in Kapitel 3.1 vorgestellte allgemeine Definition der Fragmentierung (siehe Definition 3.1) geeignet für diese Annahme spezialisiert werden. Das bedeutet in diesem Kontext, dass eine geeignete Fragmentierung $\mathcal{F}$ lediglich zwei Fragmente F_1 und F_2 enthalten darf, da ansonsten mindestens zwei Fragment-Instanzen unterschiedlicher Fragmente gemeinsam auf einem Server gespeichert werden müssten und die mögliche Kombination dieser durch einen nicht autorisierten Betrachter zu einer Verletzung von Vertraulichkeitsanforderungen führen könnte.

Da des Weiteren keine Möglichkeit besteht, einzelne Fragment-Instanzen lokal (und damit sicher vor unautorisierten Zugriffen) zu speichern, muss eine geeignete Fragmentierung $\mathcal{F} = \{F_1, F_2\}$ so beschaffen sein, dass keines der Fragmente F_1 oder F_2 für sich betrachtet ein Vertraulichkeits-Constraint missachtet. Aufgrund der Beschränkung auf lediglich zwei Fragmente kann es aber Mengen von Vertraulichkeits-Constraints geben, von denen nicht alle assoziierenden Vertraulichkeits-Constraints ausschließlich durch Fragmentierung der Datenbankrelation konfliktfrei gelöst werden können. Dazu sei ein Relationenschema mit den Attributen a_0, a_1 und a_2 betrachtet, welches unter Einhaltung der Vertraulichkeits-Constraints $c_0 = \{a_0, a_1\}$, $c_1 = \{a_1, a_2\}$ und $c_2 = \{a_2, a_0\}$ fragmentiert werden soll. Aufgrund der Tatsache, dass bei Wahl von $i \in \{0, 1, 2\}$ immer jeweils die beiden Attribute $a_{i \bmod 3}$ und $a_{(i+1) \bmod 3}$ nicht gemeinsam in einem Fragment liegen dürfen, ist es bei der gegebenen zyklischen Struktur dieser Constraints nicht möglich, allein durch Aufteilung der Menge der Attribute auf nur zwei Fragmente zwei beliebige Constraints zu erfüllen, ohne dabei das jeweils dritte Constraint zu verletzen.

Aus diesem Grund kann es passieren, dass auch bei einer Constraint-Menge, die ausschließlich assoziierende Vertraulichkeits-Constraints enthält, auf Kryptographie zurückgegriffen werden muss, um – wie auch zur Erfüllung einfacher Vertraulichkeits-Constraints – die Werte einzelner Attribute für nicht autorisierte Betrachter unlesbar zu machen. Bezogen auf das oben genannte Beispiel ist es so möglich, alle Constraints c_0, c_1 und c_2 durch Verschlüsselung der Werte eines beliebigen Attributs a_0, a_1 oder a_2 zu erfüllen.

Die Verschlüsselung der Werte von Attributen kann dabei beispielsweise durch ein symmetrisches Kryptographie-Verfahren (siehe [28, Kap. 7.5]) erfolgen, durch das ein vorkommender Attributwert[3] v mit einem individuellen kryptographischen Schlüssel k_v über eine Funktion $E(v, k_v)$ verschlüsselt wird. In der zu berechnenden Fragmentierung $\mathcal{F} = \{F_1, F_2\}$ wird dazu jedes Attribut a_j, dessen Werte verschlüsselt gespeichert werden sollen, jeweils in F_1 und in F_2 eingefügt[4]. Für das Vorkommen eines Wertes v, der für Attribut a_j in der ursprünglichen Relationeninstanz enthalten ist, wird dann in der Fragment-Instanz zu F_1 unter a_j der individuelle Schlüssel k_v abgespeichert, während das zugehörige Tupel aus der Fragment-Instanz zu F_2 Attribut a_j auf $E(v, k_v)$ abbildet.

Weiterhin muss es möglich sein, auf Basis beider Fragment-Instanzen f_1 und f_2 zu einer Fragmentierung $\mathcal{F} = \{F_1, F_2\}$ die ursprüngliche Relationeninstanz r wieder korrekt rekonstruieren zu können. Wenn die Existenz eines Primärschlüssels vorausgesetzt werden kann, der in beiden Fragmenten F_1 und F_2 enthalten ist, können die beiden Instanzen über den natürlichen Verbund $r = f_1 \bowtie f_2$ wieder korrekt zusammengesetzt werden. Es ist aber möglich, dass der Primärschlüssel eines Relationenschemas $\langle R|A_R|SC_R\rangle$ (falls ein solcher überhaupt existiert) selbst durch Vertraulichkeits-Constraints geschützt ist, so dass dieser nicht in beiden bei der Fragmentierung von $\langle R|A_R|SC_R\rangle$ entstehenden Fragmenten enthalten sein darf.

Aus diesem Grund erhalten beide Fragmente F_1 und F_2 im Rahmen der Fragmentierung jeweils zusätzlich noch eine Tupel-ID in Form des Attributs `tid`, welches die Funktion eines Primärschlüssels inne hat. In den Fragment-Instanzen f_1 und f_2 einer Fragmentierung haben dabei immer jeweils genau die beiden Tupel $\nu_1 \in f_1$ und $\nu_2 \in f_2$ denselben Wert $\nu_1[\texttt{tid}] = \nu_2[\texttt{tid}]$, die zusammengesetzt einem Tupel der ursprünglichen Relationeninstanz r entsprechen. Dabei wird durch die Vereinbarung, das Attribut `tid` in beiden Fragmenten jeweils als Primärschlüssel zu deklarieren, ausgeschlossen, dass zwei verschiedene Tupel einer Fragment-Instanz dieselbe Tupel-ID haben können.

[3] Mit v wird hier ein konkretes Vorkommen eines Wertes bezeichnet, nicht die Konstante aus der entsprechenden Domäne. Für zwei Werte v und v' mit $v = v'$ gilt deshalb trotzdem $k_v \neq k_{v'}$.

[4] Es wird davon ausgegangen, dass sowohl k_v als auch $E(v, k_v)$ jeweils binär codiert vorliegen, damit Attribut a_j in F_1 und F_2 derselbe semantische Bereichsname zugeordnet werden kann.

In Anlehnung an die Definitionen aus [2] kann damit die folgende Definition einer spezialisierten vertikalen Fragmentierung für den Ansatz „Fragmentierung und nicht kooperierende Partner“ entwickelt werden:

Definition 3.3 (Spezialisierte vertikale Fragmentierung) *Sei gemäß Definition 2.1 ein Relationenschema $\langle R|A_R|\ \ \rangle$ gegeben. Eine spezialisierte vertikale Fragmentierung dieses Relationenschemas ist durch ein Tupel $(\mathcal{F}, \mathcal{E})$ gegeben. Dabei entspricht $\mathcal{E} \subseteq A_R$ der Menge der Attribute, deren Werte verschlüsselt gespeichert werden sollen, und*

$$\mathcal{F} = \{\langle F_1|A_{F_1}|SC_{F_1}\rangle, \langle F_2|A_{F_2}|SC_{F_2}\rangle\}$$

gemäß Definition 3.1 einer vertikalen Fragmentierung von Schema $\langle R|A_R|\ \ \rangle$, in der des Weiteren für $i \in \{1, 2\}$

(i) beide $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ ein Attribut $\texttt{tid} \notin A_R$ in A_{F_i} enthalten,

(ii) jeweils $\texttt{tid} \in A_{F_i}$ als Primärschlüssel in SC_{F_i} vereinbart ist,

(iii) für beide $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ jeweils $\mathcal{E} \subseteq A_{F_i}$ Gültigkeit hat und

(iv) $(A_{F_1} \setminus \{\texttt{tid}\}) \cup (A_{F_2} \setminus \{\texttt{tid}\}) = A_R$ gilt.

Wenn r gemäß Definition 2.2 eine Relationeninstanz zu Schema $\langle R|A_R|\ \ \rangle$ ist, können die Fragment-Instanzen f_1 und f_2 zu den Fragmenten $\langle F_1|A_{F_1}|SC_{F_1}\rangle \in \mathcal{F}$ und $\langle F_2|A_{F_2}|SC_{F_2}\rangle \in \mathcal{F}$ bezüglich r gebildet werden, indem für ein Tupel $\mu \in r$ jeweils genau die beiden Tupel $\nu_1 \in f_1$ und $\nu_2 \in f_2$ existieren, wobei für $i \in \{1, 2\}$

(i) für ein Attribut $a_j \in ((A_{F_i} \cap A_R) \setminus \mathcal{E})$ entsprechend $\nu_i[a_j] = \mu[a_j]$ gilt,

(ii) für ein Attribut $a_j \in \mathcal{E}$ sowohl $\nu_1[a_j] = k_v$ als auch $\nu_2[a_j] = E(\mu[a_j], k_v)$ gilt,

(iii) für $\texttt{tid} \in A_{F_1} \cap A_{F_2}$ die Bedingung $\nu_1[\texttt{tid}] = \nu_2[\texttt{tid}]$ erfüllt wird und

(iv) für je zwei Tupel $\nu', \nu'' \in f_i$ mit $\nu' \neq \nu''$ jeweils $\nu'[\texttt{tid}] \neq \nu''[\texttt{tid}]$ gegeben ist.

Weitere Tupel existieren in f_1 und f_2 nicht.

3.2.2 Durchsetzung der Vertraulichkeitsanforderungen

Wie bereits in Kapitel 3.1.2 erläutert, erfolgt die Definition von Vertraulichkeitsanforderungen für ein gegebenes Relationenschema in Form einer wohldefinierten Menge $\mathcal{C}$ von Vertraulichkeits-Constraints. In diesem Kontext wurde auch auf die

allgemeine Semantik hinter der Idee eines Vertraulichkeits-Constraints eingegangen, die an dieser Stelle für den oben vorgestellten Ansatz der Fragmentierung bei nicht kooperierenden Partnern geeignet konkretisiert werden soll.

Definition 3.4 (Korrekte Fragmentierung) *Sei $\langle R|A_R|\ \ \rangle$ ein gemäß Definition 2.1 aufgebautes Relationenschema, für das gemäß Definition 3.2 eine wohldefinierte Menge $\mathcal{C}$ von Vertraulichkeits-Constraints gegeben ist. Eine nach Definition 3.3 gestaltete Fragmentierung $(\mathcal{F}, \mathcal{E})$ des Relationenschemas $\langle R|A_R|\ \ \rangle$ mit*

$$\mathcal{F} = \{\langle F_1|A_{F_1}|SC_{F_1}\rangle, \langle F_2|A_{F_2}|SC_{F_2}\rangle\}$$

heißt genau dann korrekt bezüglich $\mathcal{C}$, wenn für jedes Vertraulichkeits-Constraint $c_j \in \mathcal{C}$ sowohl $c_j \nsubseteq (A_{F_1} \setminus \mathcal{E})$ als auch $c_j \nsubseteq (A_{F_2} \setminus \mathcal{E})$ gilt.

Nach dieser Idee darf bei einer korrekten Fragmentierung bezüglich einer Menge $\mathcal{C}$ von Vertraulichkeits-Constraints also keines der beiden entstehenden Fragmente F_1 und F_2 eine Teilmenge $\{a_1, \ldots, a_k\}$ von nicht verschlüsselten Attributen enthalten, die in dieser Kombination durch ein Constraint $\{a_1, \ldots, a_k\} \in \mathcal{C}$ geschützt ist.

Bei einem assoziierenden Vertraulichkeits-Constraints $\{a_1, \ldots, a_k\} \in \mathcal{C}$ bedeutet dies, dass mindestens ein Attribut $a_j \in \{a_1, \ldots, a_k\}$ nicht in demselben Fragment wie die restlichen Attribute $\{a_1, \ldots, a_k\} \setminus \{a_j\}$ enthalten sein darf oder alternativ die Werte mindestens eines Attributs $a_j \in \{a_1, \ldots, a_k\}$ verschlüsselt gespeichert werden müssen. Bei Wahl der zweiten Alternative ist das gemeinsame Enthaltensein aller Attribute $\{a_1, \ldots, a_k\}$ in einem Fragment möglich. Für einen nicht autorisierten Betrachter, der nach Voraussetzung lediglich auf eine der Fragment-Instanzen f_1 oder f_2 zu den berechneten Fragmenten $F_1, F_2 \in \mathcal{F}$ zugreifen kann, ist es damit bei beiden Alternativen nicht möglich, Zugriff auf eine Assoziation konkreter Attributwerte, die durch dieses assoziierende Vertraulichkeits-Constraint geschützt wird, zu erlangen: Entweder sind die Werte mindestens eines Attributs $a_j \in \{a_1, \ldots, a_k\}$ ausschließlich in einer für ihn nicht lesbaren Fragment-Instanz enthalten oder die Werte mindestens eines Attributs $a_j \in \{a_1, \ldots, a_k\}$ sind verschlüsselt gespeichert.

Im Falle von verschlüsselt gespeicherten Attributwerten eines Attributs a_j kann ein nicht autorisierter Betrachter nach Voraussetzung nur entweder auf die verschlüsselten Attributwerte oder auf die zur Entschlüsselung der Attributwerte benötigten kryptographischen Schlüssel zugreifen. Auf die Kombination dieser beiden Daten, deren gemeinsame Kenntnis zum Entschlüsseln der verschlüsselten Werte von a_j zwingend notwendig ist, erhält er hingegen niemals Zugriff. Aus diesem Grund können auch einfache Vertraulichkeits-Constraints $\{a\} \in \mathcal{C}$, bei denen die zu a gehörenden Attributwerte selbst vertraulich sind, über eine solche Verschlüsselung der Attributwerte von a geschützt werden.

F_1	**tid**	**SSN**	**Name**	**Geburtstag**	**Plz**
	1	k_1^{SSN}	k_1^{Name}	03.01.1981	94142
	2	k_2^{SSN}	k_2^{Name}	07.10.1953	94141
	3	k_3^{SSN}	k_3^{Name}	12.02.1952	94139
	4	k_4^{SSN}	k_4^{Name}	03.01.1981	94139

F_2	**tid**	**SSN**	**Name**	**Krankheit**	**Arzt**
	1	α	ϵ	Bluthochdruck	White
	2	β	ζ	Fettleibigkeit	Warren
	3	γ	η	Bluthochdruck	White
	4	δ	θ	Fettleibigkeit	Warren

Abbildung 3.5: Fragmentierung von *patient* für nicht kooperierende Partner

Für einen autorisierten Benutzer, der Zugriff auf die Fragment-Instanzen beider Fragmente $F_1, F_2 \in \mathcal{F}$ bekommt, ist es hingegen stets möglich, alle verschlüsselten Werte wieder zu entschlüsseln. Da eine Fragmentierung eines Relationenschemas R nach Definition 3.3 außerdem garantiert, dass alle Attribute aus R in mindestens einem der beiden Fragmente F_1 oder F_2 enthalten sind, ist aus Sicht eines autorisierten Benutzers die Eigenschaft der Vollständigkeit einer Fragmentierung immer gewährleistet. Ebenso ist aus Sicht eines autorisierten Benutzers stets die Eigenschaft der Rekonstruierbarkeit erfüllt, da dieser die beiden zu einer Fragmentierung gehörenden Fragment-Instanzen f_1 und f_2 aufgrund der bereitgestellten Tupel-IDs wieder korrekt über den natürlichen Verbund $f_1 \bowtie f_2$ verknüpfen kann[5].

Ein Beispiel, wie eine Fragmentierung nach Definition 3.3 unter der Annahme nicht kooperierender Partner aussehen kann, ist in Abbildung 3.5 gegeben. In dieser sind die Fragment-Instanzen f_1 und f_2 einer Fragmentierung $\mathcal{F} = \{F_1, F_2\}$ des Relationenschemas *Patient* (siehe zugehörige Instanz in Abbildung 3.1) zu sehen, welche bezüglich der wohldefinierten Menge $\mathcal{C}$ an Vertraulichkeits-Constraints aus Abbildung 3.3 korrekt ist. Durch das einfache Constraint $c_0 \in \mathcal{C}$ wird vorgeschrieben, dass die Werte des Attributs `SSN` ausschließlich verschlüsselt gespeichert werden dürfen. Entsprechend befinden sich in f_2 ausschließlich die verschlüsselten Werte dieses Attributs, während in f_1 die zugehörigen kryptographischen Schlüssel zu finden sind. Die assoziierenden Constraints $c_1, \ldots, c_4 \in \mathcal{C}$ werden durch Verschlüsselung der Werte des Attributs `Name` geschützt und c_5 und c_6 durch Aufteilung der entsprechenden Attributmengen der Constraints auf die beiden Fragmente. Durch

[5] Im Vorfeld müssen natürlich verschlüsselte Werte in f_2 entschlüsselt und die entsprechenden Schlüssel aus f_1 über eine geeignete Projektion entfernt werden

die Werte des Primärschlüssels `tid` ist die Rekonstruierbarkeit der ursprünglichen Instanz *patient* über den natürlichen Verbund $f_1 \bowtie f_2$ gesichert.

In diesem Beispiel ist die Verschlüsselung der Werte von mindestens einem weiteren Attribut neben `SSN` notwendig, um alle Vertraulichkeits-Constraints erfüllen zu können. Da ohne den Einsatz von Verschlüsselung das Attribut `Name` aufgrund der Constraints $c_1, \ldots, c_4$ mit keinem der Attribute `Geburtstag`, `Plz`, `Krankheit` und `Arzt` gemeinsam in einem Fragment enthalten sein darf, diese Attribute aufgrund der Constraints c_5 und c_6 aber auch nicht alle gemeinsam in einem Fragment liegen dürfen, würden ohne den Einsatz von Kryptographie mindestens drei Fragmente benötigt, um alle Vertraulichkeitsanforderungen erfüllen zu können. Dies ist aber aufgrund der Annahme, nur zwei Server zu haben, auf denen jeweils eine Fragment-Instanz einer Fragmentierung gespeichert werden kann, nicht möglich.

Um Anfragen an eine fragmentierte Datenbankrelation möglichst effizient zu gestalten, kann es wünschenswert sein, dass die Werte möglichst weniger Attribute verschlüsselt gespeichert werden und ggf. bestimmte Kombinationen von Attributen möglichst zusammen in einem Fragment enthalten sind. Aufgrund dessen wird in [2, Abschnitt 5] eine Idee präsentiert, wie eine Fragmentierung (unter Einhaltung der Vertraulichkeitsanforderungen) so gestaltet werden kann, dass eine repräsentative Menge von Anfragen möglichst effizient ausgeführt werden kann.

3.3 Fragmentierung und partielle Verschlüsselung

Der in Kapitel 3.2 beschriebene Ansatz „Fragmentierung und nicht-kooperierende Partner“ erreicht Vertraulichkeitsschutz unter der Voraussetzung, dass keinerlei Kommunikation zwischen den beiden verwendeten Servern existiert und jeder nicht autorisierte Betrachter auf maximal einen der beiden Server zugreifen kann. Praktisch gesehen sind diese Annahmen aber in der Realität nur schwierig durchzusetzen: So können beispielsweise ehemals nicht kooperierende Server nach einer Fusion zweier Dienstleister unter der Kontrolle desselben Administrators stehen, der damit Zugriff auf beide Fragment-Instanzen hat und demnach – genau wie ein autorisierter Benutzer – die ursprüngliche Datenbankinstanz rekonstruieren kann.

Der in [23], [25] und [27] vorgestellte Ansatz beseitigt dieses Manko, indem eine Fragmentierung so konstruiert wird, dass alle Fragment-Instanzen gemeinsam auf demselben nicht vertrauenswürdigen Server gespeichert werden können. Dazu wird bei diesem Ansatz eine Kombination aus Fragmentierung und partieller Verschlüsselung der Daten derart eingesetzt, dass selbst bei Kenntnis aller Fragment-Instanzen der Vertraulichkeitsschutz hinsichtlich einer Menge von Vertraulichkeits-Constraints gewahrt bleibt. Wie auch bei dem Ansatz „Fragmentierung und nicht-kooperierende

Partner“ hat der verwendete Client ausschließlich die Funktion, Anfragen auszuführen, und kann insbesondere keine Fragment-Instanzen lokal speichern.

3.3.1 Fragmentierung der Datenbankrelation

Auch bei dem in diesem Kapitel behandelten Ansatz „Fragmentierung und partielle Verschlüsselung“ ist es notwendig, die allgemeine Definition der vertikalen Fragmentierung (siehe Definition 3.1 in Kapitel 3.1.1) für diesen Ansatz geeignet zu spezialisieren. Da eine Fragmentierung hier so gestaltet werden soll, dass auch die Kenntnis mehrerer Fragment-Instanzen nicht zu einer Verletzung der Vertraulichkeitsanforderungen führt, ist die Anzahl der bei der Fragmentierung entstehenden Fragmente, deren zugehörige Fragment-Instanzen alle auf demselben externen Server gespeichert werden können, bei diesem Ansatz nicht beschränkt.

Da auch bei diesem Ansatz keine Möglichkeit der lokalen (und damit vor unautorisierten Zugriffen sicheren) Speicherung einzelner Fragment-Instanzen besteht, muss eine geeignete Fragmentierung $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ auch hier so beschaffen sein, dass durch die Kenntnis einer einzelnen Fragment-Instanz keine Vertraulichkeits-Constraints verletzt werden können. Aus diesem Grund existiert zur Erfüllung einfacher Vertraulichkeits-Constraints ausschließlich die Möglichkeit, die gemäß eines solchen Constraints zu schützenden Attributwerte verschlüsselt zu speichern.

Assoziierende Vertraulichkeits-Constraints können hingegen stets durch eine geeignete Fragmentierung der Attributmenge des ursprünglichen Relationenschemas erfüllt werden, da die Anzahl der Fragmente bei dem hier vorgestellten Ansatz nicht beschränkt ist. So ist es immer möglich (wenn auch nicht wünschenswert), jedes einzelne Attribut a_j einer zu fragmentierenden Datenbankrelation, dessen Werte nicht aufgrund eines entsprechenden einfachen Vertraulichkeits-Constraints verschlüsselt gespeichert werden müssen, in einem eigenen Fragment $F_j \in \mathcal{F}$ zu verwalten. Durch dieses Vorgehen können auf triviale Art und Weise alle assoziierenden Vertraulichkeits-Constraints stets erfüllt werden. Zusätzlich können assoziierende Vertraulichkeits-Constraints der Form $\{a_1, \ldots, a_k\}$ aber auch durch Verschlüsselung der Werte mindestens eines Attributs $a_j \in \{a_1, \ldots, a_k\}$ erfüllt werden. Da dazu aber wie oben erläutert keine Notwendigkeit besteht, wird im Folgenden davon ausgegangen, dass die Erfüllung assoziierender Vertraulichkeits-Constraints ausschließlich auf Basis von Fragmentierung erfolgt.

Laut Annahme soll es möglich sein, alle Fragment-Instanzen einer fragmentierten Relationeninstanz gemeinsam auf einem nicht vertrauenswürdigen Server speichern zu können. Aufgrund dessen muss eine entsprechende Fragmentierung so beschaffen sein, dass durch Betrachtung der unverschlüsselten Werte einer beliebigen Anzahl

an Fragment-Instanzen kein assoziierendes Vertraulichkeits-Constraint verletzt werden kann. Das heißt, dass es nicht möglich sein darf, auf Basis der unverschlüsselten Attributwerte jeweils zwei beliebige verschiedene Fragment-Instanzen f_i und f_j korrekt verknüpfen zu können. Unter der Annahme,[6] dass eine solche Verknüpfung ausschließlich über den natürlichen Verbund möglich ist, wird dazu gefordert, dass jeweils zwei verschiedene Fragmente $F_i, F_j \in \mathcal{F}$ disjunkt bezüglich der in ihnen enthaltenen unverschlüsselten Attribute sind.

Die Verschlüsselung von Attributwerten soll auch bei diesem Ansatz über ein symmetrisches[7] Kryptographie-Verfahren (siehe [28, Kap. 7.5]) erfolgen, bei dem ein Wert v mit einem kryptographischen Schlüssel k über eine Funktion $E(v, k)$ verschlüsselt wird. Dabei wird bei diesem Ansatz für alle zu verschlüsselnden Attributwerte derselbe Schlüssel k gewählt. Für ein zu fragmentierendes Relationenschema $\langle R|A_R| \ \rangle$ sind dabei in einem Fragment F_i, das die unverschlüsselten Attribute $A^*_{F_i} \subseteq A_R$ enthält, alle übrigen Attribute $A_R \backslash A^*_{F_i}$ in verschlüsselter Form enthalten. Daraus resultiert, dass jede zugehörige Fragment-Instanz zu einem Fragment $F_i \in \mathcal{F}$ mit allen verschlüsselt und unverschlüsselt gespeicherten Attributwerten insgesamt die komplette Menge der Attributwerte der ursprünglichen Relationeninstanz r zu Schema R enthält. Jedes Tupel $\mu \in r$ ist damit komplett in der Fragment-Instanz f_i zu einem entsprechenden Fragment $F_i \in \mathcal{F}$ enthalten, indem ein Teil $\mu\lceil(A_R \setminus A^*_{F_i})$ von μ in f_i verschlüsselt gespeichert wird, während der andere Teil $\mu\lceil A^*_{F_i}$ dieses Tupels in f_i unverschlüsselt enthalten ist. Dabei bezeichnet $\mu\lceil A$ wie gehabt die Einschränkung eines Tupels μ auf eine Attributmenge A (siehe [7, Kap. 8.1]).

Der einfacheren Handhabbarkeit wegen wird im Folgenden davon ausgegangen, dass der Teil eines Tupels $\mu \in r$, der in einer Fragment-Instanz f_i verschlüsselt gespeichert werden soll, direkt in Form eines verschlüsselten Teil-Tupels $\mu\lceil(A_R \setminus A^*_{F_i})$ in f_i gespeichert wird. Dazu enthält das Fragment $F_i \in \mathcal{F}$, zu dem f_i gebildet wurde, das ausgezeichnete Attribut `enc`, über das in Fragment-Instanzen verschlüsselte Teil-Tupel verwaltet werden können. Dabei werden alle Teil-Tupel, die im Rahmen der Fragmentierung einer Datenbankrelation gespeichert werden, mit demselben kryptographischen Schlüssel k verschlüsselt, der ausschließlich autorisierten Benutzern zur Verfügung steht.

Da alle Teil-Tupel mit demselben Schlüssel k über dieselbe (deterministische) Funktion $E(v, k)$ verschlüsselt werden, werden zwei identische Teil-Tupel auch in verschlüsselter Form gleich repräsentiert und können deshalb auch ohne Kenntnis von k (und damit ohne Kenntnis der Klartext-Werte) als identisch erkannt werden.

[6] Die Gültigkeit dieser implizit getroffenen Annahme wird in den zugrunde liegenden Ausarbeitungen nicht näher begründet.

[7] Auch wenn in [23], [25] und [27] das eingesetzte Kryptographie-Verfahren nicht genau spezifiziert wird, scheint hier der Einsatz eines symmetrischen Kryptographie-Verfahrens angemessen.

Dies lässt innerhalb einer Fragment-Instanz f_i möglicherweise den Rückschluss zu, dass in zwei unterschiedlichen Tupeln μ und ν aus f_i die beiden unverschlüsselten Teil-Tupel $\mu\lceil A^*_{F_i}$ und $\nu\lceil A^*_{F_i}$ aufgrund ihrer identischen verschlüsselten Teil-Tupel $\mu[\texttt{enc}] = \nu[\texttt{enc}]$ in Relation stehen. Damit ist es innerhalb von f_i unter Umständen möglich, unerwünschte Verknüpfungen zwischen nicht verschlüsselten Attributwerten auf Basis verschlüsselter Werte herzustellen, obwohl aus Sicht eines nicht autorisierten Betrachters eigentlich kein Informationsgewinn auf Basis verschlüsselter Attributwerte möglich sein sollte.

Als Beispiel dazu soll eine Datenbankrelation eines Krankenhauses betrachtet werden, in der über das Attribut `Name` der Name eines Patienten und über das Attribut `Krankheit` die Krankheit eines Patienten verwaltet wird. Weitere Attribute sind in diesem Relationenschema nicht vorhanden. Die sensible Assoziation zwischen Werten dieser beiden Attribute soll per Fragmentierung geschützt werden. Aufgrund dessen existiert eine Fragment-Instanz f_i, in der die Werte von Attribut `Krankheit` verschlüsselt gespeichert werden, während die Werte von Attribut `Name` im Klartext vorliegen. Wenn nun $\mu[\texttt{enc}] = \nu[\texttt{enc}]$ für zwei Tupel $\mu, \nu \in f_i$ gilt, kann möglicherweise erschlossen werden, dass die beiden durch $\mu[\texttt{Name}]$ und $\nu[\texttt{Name}]$ identifizierten Patienten unter derselben Krankheit leiden.

Zudem ist auch eine Konstellation, in der bezogen auf denselben semantischen Kontext zwei verschiedene Relationeninstanzen r_1 und r_2 existieren, deren mit R_1 und R_2 bezeichnete Schemata jeweils eine Teilmenge an Attributen gemeinsam haben, denkbar. Falls zu r_1 eine Fragment-Instanz f_1 und zu r_2 eine Fragment-Instanz f_2 existiert und beide Fragment-Instanzen f_1 und f_2 über Attribut `enc` Teil-Tupel bezüglich derselben Menge an Attributen enthalten, können zwei Tupel $\mu \in f_1$ und $\nu \in f_2$ mit $\mu[\texttt{enc}] = \nu[\texttt{enc}]$ verknüpft werden. Innerhalb einer Relationeninstanz r ist eine solche Verknüpfung zwischen Tupeln verschiedener Fragment-Instanzen f_i und f_j über Werte von `enc` hingegen nicht möglich: Bei je zwei Fragmenten F_i und F_j, die unterschiedliche unverschlüsselte Attribute enthalten, müssen auch die verschlüsselt gespeicherten Teil-Tupel (gemäß der Vorschrift zur Bildung dieser) der zugehörigen Fragment-Instanzen unterschiedlich sein, weil sich diese nicht auf dieselbe Menge an Attributen beziehen.

Um das grundsätzliche Problem, dass identische Werte (in Form von Teil-Tupeln) auch verschlüsselt gleich repräsentiert werden, zu beheben, enthält jedes Tupel ν einer Fragment-Instanz einen sogenannten Salt (siehe [35, Kap. 3.2]) in Form einer zufällig gewählten aber eindeutigen Bitfolge, die in ν als Wert des ausgezeichneten Attributs `salt` gespeichert wird. Mit dem Salt $\nu[\texttt{salt}]$ wird die binäre Repräsentation eines in $\nu[\texttt{enc}]$ verschlüsselt zu speichernden Teil-Tupels vor der Verschlüsselung über die `XOR`-Funktion $\oplus$ binär verknüpft. Wenn alle Salt-Werte aus allen gespeicherten Fragment-Instanzen eindeutig gewählt sind, haben im Klartext identische

Teil-Tupel nach Verknüpfung mit den jeweiligen Salt-Werten (trotz Verschlüsselung mit demselben Schlüssel und über dieselbe Kryptographie-Funktion) unterschiedliche verschlüsselte Repräsentationen.

In Anlehnung an die Definitionen aus [23], [25] und [27] kann damit die folgende Definition einer spezialisierten vertikalen Fragmentierung für den Ansatz „Fragmentierung und partielle Verschlüsselung“ entwickelt werden:

Definition 3.5 (Spezialisierte vertikale Fragmentierung) *Sei gemäß Definition 2.1 ein Relationenschema $\langle R|A_R|\ \ \rangle$ gegeben. Eine spezialisierte vertikale Fragmentierung dieses Relationenschemas ist durch eine Fragmentierung*

$$\mathcal{F} = \{\langle F_1|A_{F_1}|SC_{F_1}\rangle, \langle F_2|A_{F_2}|SC_{F_2}\rangle, \ldots, \langle F_n|A_{F_n}|SC_{F_n}\rangle\}$$

gemäß Definition 3.1 gegeben, in der des Weiteren für $i \in \{1, \ldots, n\}$

(i) für jedes $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ ein Attribut $\texttt{enc} \notin A_R$ in A_{F_i} enthalten ist,

(ii) für jedes $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ ein Attribut $\texttt{salt} \notin A_R$ in A_{F_i} enthalten ist und $\texttt{salt} \in A_{F_i}$ jeweils als Primärschlüssel in SC_{F_i} vereinbart ist und

(iii) für je zwei verschiedene Fragmente $\langle F_i|A_{F_i}|SC_{F_i}\rangle, \langle F_j|A_{F_j}|SC_{F_j}\rangle \in \mathcal{F}$ jeweils immer $\left(A_{F_i} \cap A_{F_j}\right) \cap A_R = \emptyset$ gilt.

Wenn r gemäß Definition 2.2 eine Relationeninstanz zu Schema $\langle R|A_R|\ \ \rangle$ ist, kann eine Fragment-Instanz f_i zu Fragment $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ bezüglich r gebildet werden, indem für ein Tupel $\mu \in r$ jeweils genau ein Tupel $\nu \in f_i$ existiert, wobei

(i) für ein Attribut $a_j \in (A_{F_i} \cap A_R)$ entsprechend $\nu[a_j] = \mu[a_j]$ gilt,

(ii) $\nu[\texttt{enc}] = E(\mu\lceil(A_R \setminus A_{F_i}) \oplus \nu[\texttt{salt}], k)$ Gültigkeit hat und

(iii) für ein beliebiges Tupel $\nu \in f_i$ und ein beliebiges Tupel $\nu' \in f_j$ mit $\nu' \neq \nu$ jeweils $\nu[\texttt{salt}] \neq \nu'[\texttt{salt}]$ gegeben ist, wenn f_j eine beliebige global existierende Fragment-Instanz ist, für die auch $f_j = f_i$ gelten darf.

Weitere Tupel existieren in f_i nicht.

Wenn alle assoziierenden Vertraulichkeits-Constraints einer wohldefinierten Menge $\mathcal{C}$ von Vertraulichkeits-Constraints grundsätzlich mit Hilfe von Fragmentierung (anstelle von Verschlüsselung) erfüllt werden, gilt in einer Fragmentierung nach Definition 3.5 zusätzlich, dass für jedes Attribut $a_j \in A_R$, welches nicht in einem einfachen Vertraulichkeits-Constraint $\{a_j\} \in \mathcal{C}$ vorkommt, genau ein Fragment $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ mit $a_j \in A_{F_i}$ existiert.

3.3.2 Durchsetzung der Vertraulichkeitsanforderungen

Auch für den in diesem Kapitel vorgestellten Ansatz „Fragmentierung und partielle Verschlüsselung“ ist es notwendig, die konkrete Semantik einer Menge wohldefinierter Vertraulichkeits-Constraints, über die gewünschte Vertraulichkeitsanforderungen ausgedrückt werden, zu definieren.[8]

Definition 3.6 (Korrekte Fragmentierung) *Sei $\langle R|A_R|\ \ \rangle$ ein gemäß Definition 2.1 aufgebautes Relationenschema, für das gemäß Definition 3.2 eine wohldefinierte Menge $\mathcal{C}$ von Vertraulichkeits-Constraints gegeben ist. Eine nach Definition 3.5 gestaltete Fragmentierung $\mathcal{F}$ des Relationenschemas $\langle R|A_R|\ \ \rangle$ heißt genau dann korrekt bezüglich $\mathcal{C}$, wenn für jedes Vertraulichkeits-Constraint $c_j \in \mathcal{C}$ und für jedes Fragment $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ jeweils $c_j \nsubseteq A_{F_i}$ gilt.*

Gemäß dieser Definition ist eine Fragmentierung also korrekt bezüglich einer Menge $\mathcal{C}$ von Vertraulichkeits-Constraints, wenn in keinem Fragment $F_i \in \mathcal{F}$ alle Attribute eines beliebigen Vertraulichkeits-Constraints $c_j \in \mathcal{C}$ als Teilmenge enthalten sind. Diese Eigenschaft wird dabei implizit ausschließlich für Attribute, deren Werte unverschlüsselt gespeichert werden sollen, gefordert, weil nach Definition 3.2 ein Vertraulichkeits-Constraint $c_j \in \mathcal{C}$ ausschließlich Attribute des ursprünglichen Relationenschemas R enthalten darf und damit insbesondere die beiden ausgezeichneten Attribute `salt` und `enc` nicht in c_j vorkommen können (vgl. Definition 3.5).

Im Falle eines assoziierenden Vertraulichkeits-Constraints $\{a_1, \ldots, a_k\} \in \mathcal{C}$ darf deshalb – unabhängig davon, ob dieses Constraint durch Fragmentierung oder Verschlüsselung erfüllt wird – in jedem Fragment $F_i \in \mathcal{F}$ immer mindestens ein Attribut $a_j \in \{a_1, \ldots, a_k\}$ ausschließlich in verschlüsselter Form enthalten sein. Somit können aus der Sichtweise eines nicht autorisierten Betrachters – dem der zur Entschlüsselung der (verschlüsselt gespeicherten) Teil-Tupel benötigte kryptographische Schlüssel k nicht bekannt ist – die Werte dieses Attributs a_j auf Basis einer einzelnen Fragment-Instanz f_i nicht mit den Werten der restlichen Attribute $\{a_1, \ldots, a_k\} \setminus \{a_j\}$ dieses Constraints verknüpft werden.

Dabei ist die isolierte Betrachtung einer einzelnen Fragment-Instanz hier ausreichend, um die Erfüllung assoziierender Vertraulichkeits-Constraints zu begründen: Ein nicht autorisierter Betrachter hat zwar eventuell Zugriff auf alle Fragment-Instanzen, kann diese aber – unter der getroffenen Annahme, dass eine korrekte

[8] Im Gegensatz zu [23, 27] wird die Forderung nach Disjunktheit der Attributmengen der Fragmente hier (wie auch in [25]) bereits im Rahmen der Konstruktion der Fragmentierung gefordert (vgl. Definition 3.5), um zwischen der Konstruktion einer Fragmentierung und der korrekten Durchsetzung der Vertraulichkeitsanforderungen für diese Fragmentierung zu differenzieren.

Verknüpfung von Fragment-Instanzen ausschließlich durch den natürlichen Verbund möglich ist – ohne Kenntnis des kryptographischen Schlüssels k nicht korrekt zusammensetzen, weil nach Konstruktion der Fragmentierung jedes Attribut des ursprünglichen Relationenschemas in maximal einem Fragment in unverschlüsselter Form vorkommen darf.

Die Werte eines einfachen Vertraulichkeits-Constraints $\{a\} \in \mathcal{C}$ können nach Definition 3.6 ausschließlich durch Verschlüsselung geschützt werden, weil die gemäß dieser Definition geforderte Bedingung $\{a\} \nsubseteq A_{F_i}$ verletzt würde, wenn Attribut a unverschlüsselt in einem beliebigen Fragment $\langle F_i | A_{F_i} | SC_{F_i} \rangle \in \mathcal{F}$ enthalten wäre. Deshalb kann ein nicht autorisierter Betrachter also auch Werte, die durch einfache Vertraulichkeits-Constraints geschützt sind, mangels des Wissens über den verwendeten kryptographischen Schlüssel k nicht einsehen.

Einem autorisierten Benutzer, dem der verwendete kryptographische Schlüssel k bekannt ist, ist es hingegen stets möglich, verschlüsselte Werte einzusehen. In jeder Fragment-Instanz f_i, die bei der Fragmentierung einer ursprünglichen Relationeninstanz r entstanden ist, sind nach Entschlüsselung der verschlüsselten Teil-Tupel alle Tupel aus r vollständig enthalten. Deshalb gelten aus Sicht eines autorisierten Benutzers trivialerweise stets die Eigenschaften der Vollständigkeit und der Rekonstruierbarkeit der Fragmentierung. Aus diesem Grund ist es stets möglich, jede Anfrage, die über die ursprüngliche Instanz r beantwortet werden kann, mit Hilfe einer einzelnen Fragment-Instanz von r zu beantworten. Es ist aber meist dennoch sinnvoll, alle bei der Fragmentierung entstandenen Fragment-Instanzen auf externen Servern zu speichern, um zur effizienten Beantwortung möglichst vieler Anfragen jeweils eine Fragment-Instanz wählen zu können, mit der die jeweilige Anfrage ohne Entschlüsselung von Teil-Tupeln beantwortet werden kann.

Ein Beispiel, wie eine Fragmentierung nach Definition 3.5 unter Verwendung partieller Verschlüsselung aussehen kann, ist in Abbildung 3.6 gegeben. In dieser sind die Fragment-Instanzen f_1, f_2 und f_3 einer Fragmentierung $\mathcal{F} = \{F_1, F_2, F_3\}$ des Relationenschemas *Patient* (siehe zugehörige Instanz in Abbildung 3.1) zu sehen, welche bezüglich der wohldefinierten Menge $\mathcal{C}$ an Vertraulichkeits-Constraints aus Abbildung 3.3 korrekt ist.

In diesem Beispiel werden alle assoziierenden Vertraulichkeits-Constraints mit Hilfe von Fragmentierung gelöst. Dazu werden mindestens drei Fragmente benötigt, weil das Attribut `Name` aufgrund der Constraints $c_1, \ldots, c_4$ nicht gemeinsam mit den Attributen `Geburtstag`, `Plz`, `Krankheit` und `Arzt` unverschlüsselt in einem Fragment enthalten sein darf. Diese Attribute dürfen aufgrund der Constraints c_5 und c_6 aber wiederum auch nicht alle gemeinsam (unverschlüsselt) in einem Fragment liegen,

F_1	salt	enc	Name
	s_1	α	Hellmann
	s_2	β	Dooley
	s_3	γ	McKinley
	s_4	δ	Ripley

F_2	salt	enc	Geburtstag	Plz
	s_5	ϵ	03.01.1981	94142
	s_6	ζ	07.10.1953	94141
	s_7	η	12.02.1952	94139
	s_8	θ	03.01.1981	94139

F_3	salt	enc	Krankheit	Arzt
	s_9	ϑ	Bluthochdruck	White
	s_{10}	ι	Fettleibigkeit	Warren
	s_{11}	κ	Bluthochdruck	White
	s_{12}	λ	Fettleibigkeit	Warren

Abbildung 3.6: Fragmentierung von *patient* bei partieller Verschlüsselung

so dass ein drittes Fragment benötigt wird. Lediglich das einfache Vertraulichkeits-Constraint $c_0 = \{\texttt{SSN}\}$ wird über Verschlüsselung erfüllt. Aufgrund dessen sind die Werte des Attributs `SSN` in keiner Fragment-Instanz im Klartext enthalten.

Ziel muss es sein, eine Fragmentierung so zu gestalten, dass diese neben der Erfüllung aller Vertraulichkeits-Constraints auch eine möglichst effiziente Beantwortung von Anfragen ermöglicht. Dazu ist es üblicherweise erstrebenswert, dass möglichst wenige Fragmente benötigt werden, um assoziierende Vertraulichkeits-Constraints ohne Verwendung von Verschlüsselung zu erfüllen. Dadurch sollen möglichst viele Attributwerte und die Assoziationen zwischen diesen in den einzelnen Fragment-Instanzen im Klartext erhalten bleiben. Ansätze dazu werden in [27, Abschnitt 5] und [25, Abschnitte 3+4] vorgestellt. In [27, Abschnitt 6] wird zudem eine Möglichkeit diskutiert, explizite Verfügbarkeitsanforderungen zu modellieren, indem vorgegeben wird, welche unverschlüsselten Attribute sich nach Möglichkeit zusammen in einem Fragment befinden sollen (vgl. Kapitel 3.1.3).

3.4 Fragmentierung und partielle lokale Verwaltung

Die bislang in den Kapiteln 3.2 und 3.3 vorgestellten Ansätze basieren beide auf der Idee, dass Vertraulichkeitsanforderungen durch eine kombinierte Anwendung von Fragmentierung und Verschlüsselung durchgesetzt werden. Im Folgenden soll hier der in [24] und [26] entwickelte Ansatz vorgestellt werden, in dem ausschließlich Fragmentierung zur Durchsetzung gewünschter Vertraulichkeitsanforderungen zum Einsatz kommt und nicht auf kryptographische Verfahren zurückgegriffen werden

braucht. Im Gegensatz zu den beiden bisher vorgestellten Ansätzen ist es dafür möglich, Teile des Datenbestandes auch lokal auf dem vertrauenswürdigen Client zu speichern. Zu diesem Zweck muss der verwendete Client ausreichend Speicher und ein geeignetes relationales Datenbanksystem zur Verfügung stellen.

3.4.1 Fragmentierung der Datenbankrelation

Auch für den hier behandelten Ansatz „Fragmentierung und partielle lokale Verwaltung“ muss die allgemeine Definition der vertikalen Fragmentierung aus Definition 3.1 an die gegebenen Voraussetzungen angepasst werden. Dabei muss beachtet werden, dass in diesem Ansatz neben einem nicht vertrauenswürdigen Server eines externen Dienstleisters auch der Client, der als vollkommen vertrauenswürdig gilt (vgl. Kapitel 3.1.3), zur Speicherung von Fragmenten genutzt werden kann. Aus diesem Grund wird hier davon ausgegangen, dass eine Fragmentierung die Form $\mathcal{F} = \{F_o, F_s\}$ hat. Dabei entspricht $F_o \in \mathcal{F}$ einem Fragment, dessen zugehörige Fragment-Instanzen ausschließlich auf dem vertrauenswürdigen Client gespeichert werden dürfen, während Fragment-Instanzen zu Fragment $F_s \in \mathcal{F}$ auch bedenkenlos auf einem kostengünstigeren externen Server ausgelagert werden können.

Damit können einfache Vertraulichkeits-Constraints, welche die zu einem Attribut gehörenden Werte schützen, im Gegensatz zu den beiden bisher vorgestellten Verfahren ohne den Einsatz von Verschlüsselung erfüllt werden, indem die Werte entsprechender Attribute ausschließlich auf dem vertrauenswürdigen Client gespeichert werden. Trotz der Beschränkung der Fragmentierung auf lediglich zwei Fragmente F_o und F_s ist es hier des Weiteren immer möglich, jede zulässige Menge assoziierender Vertraulichkeits-Constraints ohne den Einsatz von Verschlüsselung zu erfüllen. Diese Möglichkeit besteht, weil auf dem Client gehaltene Datenbestände nach Voraussetzung sicher vor unbefugtem Zugriff sind und damit lediglich bei Speicherung von Attributwerten auf dem externen Server darauf geachtet werden muss, dass keine Vertraulichkeits-Constraints verletzt werden. So ist es bei jeder möglichen Menge assoziierender Vertraulichkeits-Constraints grundsätzlich immer möglich (wenn auch aus Kostengründen nicht wünschenswert), alle Constraints dieses Typs zu erfüllen, indem die Werte aller entsprechenden Attribute ausschließlich lokal auf dem Client gespeichert werden.

Zudem soll auch bei diesem Ansatz im Sinne der Eigenschaft der Rekonstruierbarkeit sichergestellt werden, dass eine ursprüngliche Relationeninstanz r, die im Zuge der Fragmentierung in die Fragment-Instanzen f_o und f_s zerlegt wird, wieder korrekt über den natürlichen Verbund $f_o \bowtie f_s$ zusammengesetzt werden kann. Dazu erhält jedes der beiden Fragmente F_o und F_s analog zu der Vorgehensweise des Ansatzes „Fragmentierung und nicht kooperierende Partner“ (vgl. Kapitel 3.2.1) ein

zusätzliches Attribut in Form des Primärschlüssels `tid`. Des Weiteren soll auch im Sinne der Eigenschaft der Vollständigkeit garantiert werden, dass jedes Attribut des ursprünglichem Relationenschemas in mindestens einem der beiden Fragmente F_o oder F_s enthalten ist.

In Anlehnung an die Definitionen aus [24] und [26] kann damit die folgende Definition einer spezialisierten vertikalen Fragmentierung für den Ansatz „Fragmentierung und partielle lokale Verwaltung“ entwickelt werden:

Definition 3.7 (Spezialisierte vertikale Fragmentierung) *Sei gemäß Definition 2.1 ein Relationenschema $\langle R|A_R|\ \ \rangle$ gegeben. Eine spezialisierte vertikale Fragmentierung dieses Relationenschemas ist durch eine Fragmentierung*

$$\mathcal{F} = \{\langle F_o|A_{F_o}|SC_{F_o}\rangle, \langle F_s|A_{F_s}|SC_{F_s}\rangle\}$$

gemäß Definition 3.1 gegeben, in der des Weiteren für $i \in \{o, s\}$

(i) beide $\langle F_i|A_{F_i}|SC_{F_i}\rangle \in \mathcal{F}$ ein Attribut $\texttt{tid} \notin A_R$ in A_{F_i} enthalten,

(ii) jeweils $\texttt{tid} \in A_{F_i}$ Primärschlüssel in SC_{F_i} ist und

(iii) $(A_{F_o} \setminus \{\texttt{tid}\}) \cup (A_{F_s} \setminus \{\texttt{tid}\}) = A_R$ gilt.

Wenn r gemäß Definition 2.2 eine Relationeninstanz zu Schema $\langle R|A_R|\ \ \rangle$ ist, können die Fragment-Instanzen f_o und f_s zu den Fragmenten $F_o \in \mathcal{F}$ und $F_s \in \mathcal{F}$ bezüglich r gebildet werden, indem für ein Tupel $\mu \in r$ jeweils genau die Tupel $\nu_o \in f_o$ und $\nu_s \in f_s$ existieren, wobei für $i \in \{o, s\}$

(i) für ein Attribut $a_j \in (A_{F_i} \cap A_R)$ entsprechend $\nu_i[a_j] = \mu[a_j]$ gilt,

(ii) für $\texttt{tid} \in A_{F_o} \cap A_{F_s}$ die Bedingung $\nu_o[\texttt{tid}] = \nu_s[\texttt{tid}]$ erfüllt wird und

(iii) für je zwei Tupel $\nu', \nu'' \in f_i$ mit $\nu' \neq \nu''$ jeweils $\nu'[\texttt{tid}] \neq \nu''[\texttt{tid}]$ gegeben ist.

Weitere Tupel existieren in f_o und f_s nicht.

Zusätzlich kann gefordert werden, dass bei einer Fragmentierung die Bedingung $(A_{F_o} \cap A_{F_s}) \cap A_R = \emptyset$ erfüllt sein soll. Damit wird die Eigenschaft der Disjunktheit (entspricht größtmöglicher Überlappungsminimalität) einer Fragmentierung gefordert, um die nicht notwendige doppelte Speicherung von Werten bestimmter Attribute zu unterbinden und damit insbesondere auf Seite des Clients den benötigten Speicherplatz nicht unnötig in die Höhe zu treiben.

3.4.2 Durchsetzung der Vertraulichkeitsanforderungen

Auch für den in diesem Kapitel vorgestellten Ansatz „Fragmentierung und partielle lokale Verwaltung“ ist es notwendig, die konkrete Semantik einer Menge wohldefinierter Vertraulichkeits-Constraints genau zu definieren.[9]

Definition 3.8 (Korrekte Fragmentierung) *Sei $\langle R|A_R|\ \ \rangle$ ein gemäß Definition 2.1 aufgebautes Relationenschema, für das gemäß Definition 3.2 eine wohldefinierte Menge $\mathcal{C}$ von Vertraulichkeits-Constraints gegeben ist. Eine nach Definition 3.7 gestaltete Fragmentierung $\mathcal{F}$ des Relationenschemas $\langle R|A_R|\ \ \rangle$ mit*

$$\mathcal{F} = \{\langle F_o|A_{F_o}|SC_{F_o}\rangle, \langle F_s|A_{F_s}|SC_{F_s}\rangle\}$$

heißt genau dann korrekt bezüglich $\mathcal{C}$, wenn für jedes Vertraulichkeits-Constraint $c_j \in \mathcal{C}$ jeweils $c_j \nsubseteq A_{F_s}$ gilt.

Nach dieser Definition muss also bei der Berechnung einer korrekten Fragmentierung für den Ansatz „Fragmentierung und partielle lokale Verwaltung“ verhindert werden, dass die in einem beliebigen Vertraulichkeits-Constraint enthaltene Attributmenge komplett in Fragment F_s, dessen zugehörige Fragment-Instanzen extern gespeichert werden können, enthalten ist. Für die Konstruktion des Fragments F_o, dessen Fragment-Instanzen ausschließlich auf dem vertrauenswürdigen Client gespeichert werden dürfen, gibt es diesbezüglich keine Einschränkungen.

Aus Definition 3.8 folgt unmittelbar, dass die Werte eines Attributs a, welche durch ein einfaches Vertraulichkeits-Constraints $\{a\} \in \mathcal{C}$ geschützt sind, nicht auf einem externen Server gespeichert werden dürfen, weil Attribut a nicht in Fragment F_s enthalten sein darf. Damit dürfen diese Werte nur lokal auf dem Client gespeichert werden und ein nicht autorisierter Betrachter kann nach Voraussetzung keinen Zugriff auf diese Werte bekommen.

Bei einem assoziierenden Vertraulichkeits-Constraint der Form $\{a_1, \ldots, a_k\}$ wird dabei von einer Fragmentierung $\mathcal{F} = \{F_o, F_s\}$ gefordert, dass mindestens ein Attribut $a_j \in \{a_1, \ldots, a_k\}$ ausschließlich in Fragment F_o enthalten sein darf. Durch diese Forderung wird gewährleistet, dass die Werte mindestens eines Attributs aus jedem

[9] Im Gegensatz zu [24, 26] wird die Forderung, dass jedes Attribut des Schemas der urspünglichen Relationeninstanz in mindestens einem Fragment vorkommen muss, hier bereits im Rahmen der Konstruktion der Fragmentierung gefordert (vgl. Definition 3.7), um zwischen der Konstruktion einer Fragmentierung und der korrekten Durchsetzung der Vertraulichkeitsanforderungen für diese Fragmentierung zu differenzieren. Die in [26, Definition 2] zusätzlich geforderte Disjunktheit der Attributmengen der Fragmente wird hier im Rahmen der Konstruktion als optional erwähnt, da diese hier für die Sicherheit nicht von Relevanz ist.

F_o	**tid**	**SSN**	**Name**	**Geburtstag**
	1	12345	Hellmann	03.01.1981
	2	98765	Dooley	07.10.1953
	3	24689	McKinley	12.02.1952
	4	13579	Ripley	03.01.1981

F_s	**tid**	**Plz**	**Krankheit**	**Arzt**
	1	94142	Bluthochdruck	White
	2	94141	Fettleibigkeit	Warren
	3	94139	Bluthochdruck	White
	4	94139	Fettleibigkeit	Warren

Abbildung 3.7: Fragmentierung von *patient* bei partieller lokaler Verwaltung

assoziierenden Vertraulichkeits-Constraints für einen nicht autorisierten Betrachter nicht einsehbar sind. Aus dessen Sichtweise kann damit auch die zu schützende Assoziation zwischen den Werten der Attribute eines assoziierenden Vertraulichkeits-Constraints nicht aufgedeckt werden.

Aus Sicht eines autorisierten Benutzers, der nach Voraussetzung auch Zugriff auf lokal gespeicherte Fragment-Instanzen hat, sind hingegen auch bei diesem Ansatz stets die Eigenschaften der Vollständigkeit und der Rekonstruierbarkeit der berechneten Fragmentierung gewährleistet. Vollständigkeit ist stets gegebenen, weil nach Konstruktion einer Fragmentierung gemäß Definition 3.7 jedes Attribut in mindestens einem Fragment F_o oder F_s enthalten sein muss. Rekonstruierbarkeit ist durch die in den Fragment-Instanzen enthalten eindeutigen Tupel-IDs garantiert: Diese ermöglichen bezogen auf zwei Fragment-Instanzen f_o und f_s einer Fragmentierung $\mathcal{F} = \{F_o, F_s\}$ jeweils die Kombination genau der beiden Tupel $\nu_o \in f_o$ und $\nu_s \in f_s$ mit $\nu_o[\texttt{tid}] = \nu_s[\texttt{tid}]$, die zusammengenommen (nach Entfernen von Attribut `tid`) einem Tupel der ursprünglichen Relationeninstanz entsprechen.

Ein Beispiel, wie eine Fragmentierung nach Definition 3.7, bei der die zu $F_o \in \mathcal{F}$ gebildete Fragment-Instanz f_o lokal auf dem vertrauenswürdigen Client gespeichert wird, aussehen kann, ist in Abbildung 3.7 gegeben. In dieser sind die Fragment-Instanzen f_o und f_s einer Fragmentierung $\mathcal{F} = \{F_o, F_s\}$ des Relationenschemas *Patient* (siehe zugehörige Instanz in Abbildung 3.1) zu sehen. Dabei ist die berechnete Fragmentierung korrekt bezüglich der in Abbildung 3.3 zu sehenden wohldefinierten Menge $\mathcal{C}$ von Vertraulichkeits-Constraints. In diesem Beispiel wird durch die ausschließlich lokale Speicherung der Werte des Attributs `SSN` sichergestellt, dass das einfache Vertraulichkeits-Constraint $c_0 = \{\texttt{SSN}\} \in \mathcal{C}$ erfüllt wird.

Da Attribut `Name` ausschließlich in F_o enthalten ist, sind für jedes der assoziierenden Vertraulichkeits-Constraints $c_1, \ldots, c_4 \in \mathcal{C}$ aus Sicht eines nicht autorisierten Betrachters die Werte mindestens eines Attributs dieses Constraints nicht einsehbar. Aufgrund dessen werden allein durch die Zuordnung von Attribut `Name` zu Fragment F_o alle Constraints $c_1, \ldots, c_4$ erfüllt. Ebenso werden in diesem Beispiel auch durch Zuordnung des Attributs `Geburtstag` zu Fragment F_o die assoziierenden Vertraulichkeits-Constraints c_5 und c_6 der Menge $\mathcal{C}$ korrekt beachtet, weil ein nicht autorisierter Betrachter damit jeweils auf die Werte mindestens eines Attributs aus c_5 und c_6 nicht zugreifen kann.

Ziel beim Berechnen einer derart gestalteten Fragmentierung muss es sein, neben der Beachtung aller definierten Vertraulichkeits-Constraints möglichst wenig Speicherplatz auf dem lokalen Client zu verbrauchen, da der Speicherplatz auf diesem nach Annahme deutlich kostenintensiver als der Speicherplatz auf dem verwendeten externen Server ist. Alternativ kann auch der Wunsch existieren, die zu berechnende Fragmentierung derart zu gestalten, dass bei einer repräsentativen Menge angenommener Anfragen an den Client möglichst viele der notwendigen Berechnungen vom Server übernommen werden können, der nach Annahme wesentlich mehr Rechenleistung als der Client zur Verfügung stellen kann. Ohne ein solches Optimierungsziel wäre es bei diesem Ansatz immer möglich, eine korrekte Fragmentierung zu erzeugen, indem alle Attribute des ursprünglichen Relationenschemas R ausschließlich Fragment F_o zugeordnet werden. Möglichkeiten zur Modellierung dieser Optimierungsziele werden in [26, Abschnitt 4] diskutiert.

4 Kontrollierte Anfrageauswertung

In Kapitel 1.1.2 wird im Kontext der Durchsetzung von Schutzzielen auf gängige Konzepte zur Zugriffskontrolle eingegangen. Dabei wird bezogen auf Vertraulichkeitsanforderungen angemerkt, dass durch Zugriffskontroll-Mechanismen zwar effektiv der Zugriff eines Benutzers auf einzelne Daten unterbunden werden kann, aber in der Regel kein Schutz vor möglichen Inferenzen erreicht wird. Wenn ein System inferenzsicher gestaltet werden soll, müssen Verfahren zum Einsatz kommen, die mögliche Inferenzen eines Benutzers erkennen und unterbinden können.

Diese Anforderung der Inferenzsicherheit erfüllen jeweils die einzelnen Verfahren der sogenannten kontrollierten Anfrageauswertung (Controlled Query Evaluation). Unter diesem Oberbegriff wird eine Vielzahl konkreter Verfahren zusammengefasst, die aber im Wesentlichen alle auf den gleichen Klassen von Komponenten und auf denselben grundsätzlichen Ideen zur inferenzsicheren Auswertung von Anfragen eines Benutzers basieren. Aufgrund dessen soll im Folgenden dieses Kapitels nicht detailliert auf einzelne existierende Verfahren eingegangen werden, sondern es sollen die wesentlichen Klassen von Komponenten und die grundsätzlichen Ideen zur inferenzsicheren Auswertung von Anfragen vorgestellt werden.

4.1 Komponenten der kontrollierten Anfrageauswertung

Allen Verfahren der kontrollierten Anfrageauswertung ist gemein, dass sie auf drei wesentlichen Komponenten basieren. Um die kontrollierte Auswertung von Anfragen eines Benutzers durchführen zu können, muss stets eine (logik-orientierte) Datenbankinstanz, eine Vertraulichkeitspolitik und eine Modellierung des (erwarteten) Wissens dieses Benutzers existieren. Diese drei Komponenten sollen im Folgenden dieses Kapitels vorgestellt werden.

4.1.1 Logik-orientierte Datenbankinstanz

Um Anfragen eines Benutzers beantworten zu können, ist eine Ansammlung von Wissen erforderlich. Dieses Wissen kann als Information, die in Form von Daten

vorliegt, betrachtet werden. Zur strukturierten Verwaltung dieser Daten bieten sich die aus Kapitel 2 bekannten Konzepte der relationalen Datenbanktheorie an, nach der Daten in relationalen Datenbankinstanzen gespeichert werden können.

Da es das erklärte Ziel ist, das Wissen, das in einer solchen Datenbankinstanz verwaltet wird, durch Mechanismen der Inferenzkontrolle zu schützen, ist es zwingend erforderlich, die Inhalte einer solchen Datenbankinstanz nicht nur als reine Daten zu betrachten. Darüber hinaus muss die Information, die durch diese Daten repräsentiert wird, ausgewertet werden können (vgl. [10, Abschnitt 1]). Andernfalls wäre es nicht möglich, die Inferenzen zu berechnen, die einem bestimmten Benutzer ermöglicht werden, indem dieser die Information, die sich aus seinen Anfragen ergibt, auf semantischer Ebene verarbeitet (vgl. [10, Abschnitt 4]). Um die Betrachtung der Inhalte einer Datenbankinstanz als Information zu ermöglichen, sollen im Rahmen der Ansätze der kontrollierten Anfrageauswertung Datenbankinstanzen in einer logik-orientierten Darstellung zum Einsatz kommen.

Voraussetzung für eine solche logik-orientierte Darstellung ist die Existenz eines konkreten logischen Systems, in dem sich die gewünschte logik-orientierte Darstellung realisieren lässt. Wie aus Kapitel 2.1 bekannt ist, wird eine relationale Datenbankinstanz zu einem bestimmten relationalen Datenbankschema gebildet. Dabei kann das relationale Datenbankschema auch als die Menge von Datenbankinstanzen aufgefasst werden, die auf der Grundlage dieses Schemas gebildet werden können (vgl. [7, Kap. 8.1]). Für die logik-orientierte Darstellung einer solchen Datenbankinstanz muss deshalb eine logische Sprache gewählt werden, deren Syntax dazu geeignet ist, jede relationale Datenbankinstanz, die sich aufgrund des betrachteten Datenbankschemas konstruieren lässt, auch in der logik-orientierten Darstellung zu modellieren (vgl. [19, Abschnitt 2]).

Neben der Syntax muss auch eine geeignete Semantik für die betrachtete logische Sprache definiert werden, um entscheiden zu können, wann eine Menge von Formeln dieser Sprache erfüllt ist. Dazu muss eine geeignete Interpretation für diese Sprache gefunden werden, in der sich – ebenso wie in der Syntax der betrachteten logischen Sprache – die Eigenschaften relationaler Datenbankinstanzen widerspiegeln (vgl. [14, Abschnitt 2]). Zu einer solchen Interpretation muss zudem auch ein geeigneter Implikationsbegriff gefunden werden, durch den innerhalb des verwendeten logischen Systems eindeutig entschieden werden kann, ob eine Formel aus der betrachteten Sprache durch eine Menge von Formeln dieser Sprache impliziert wird. Auf Basis dieses Implikationsbegriffs lässt sich dann entscheiden, welche Informations-Aspekte sich (innerhalb des betrachteten logischen Systems) aus einer gegebenen Menge von Informations-Aspekten erschließen lassen. Auf diese Weise können mögliche Inferenzen eines Benutzers mit Hilfe dieses Implikationsbegriffs bestimmt werden [19, Abschnitt 2].

Eine mögliche konkrete Ausgestaltung einer solchen logischen Sprache wird beispielsweise in Kapitel 5.1.1 diskutiert. Dort wird eine bestimmte prädikatenlogische Sprache definiert, für die in Definition 5.1 eine passende Semantik durch eine sogenannte DB-Interpretation vorgeschlagen wird. In diesem Rahmen wird in Definition 5.2 auch ein dazu passender Implikationsbegriff erläutert.

Die konkrete Darstellung einer bestimmten Datenbankinstanz innerhalb einer logischen Sprache, die zu dem betrachteten Datenbankschema passend konstruiert ist, ist davon abhängig, ob eine vollständige oder eine unvollständige Datenbankinstanz repräsentiert werden soll [10, Abschnitt 5]. Vollständige Datenbankinstanzen sind dadurch charakterisiert, dass sie (innerhalb des betrachteten logischen Systems) vollständiges Wissen haben. Deshalb können diese jeden Informations-Aspekt, der in der betrachteten logischen Sprache ausgedrückt werden kann, eindeutig als wahr oder als nicht wahr beantworten.

Aufgrund dessen kann eine vollständige Datenbankinstanz repräsentiert werden, indem ausschließlich alles positiv geltende Wissen als solches explizit modelliert wird. Negatives Wissen wird hingegen implizit behandelt, indem es nicht in dieser Datenbankinstanz enthalten ist. Diese Semantik vollständiger Datenbankinstanzen deckt sich dabei mit den gängigen Semantiken für logische Systeme: Wenn eine Interpretation eine Formel einer logischen Sprache nicht erfüllt, erfüllt sie gemäß der Auswertung einer Formel über ihren induktiven Aufbau deren Negation (siehe [32, Kap. I.3]). Aufgrund dieser Übereinstimmung der Semantiken kann eine vollständige Datenbankinstanz in der logik-orientierten Sichtweise als eine Interpretation der betrachteten logischen Sprache aufgefasst werden, die durch die betrachtete Datenbankinstanz induziert wird [19, Abschnitt 2].

Im Gegensatz zu vollständigen Datenbankinstanzen kann es im Falle von unvollständigen Datenbankinstanzen vorkommen, dass für einen bestimmten Informations-Aspekt, der durch die zugrunde liegende logische Sprache ausgedrückt werden kann, nicht definiert ist, ob dieser in der betrachteten Datenbankinstanz gilt oder nicht. Aufgrund dessen wird eine solche unvollständige Datenbankinstanz in der logik-orientierten Darstellung durch eine Menge von (geschlossenen) Formeln der verwendeten logischen Sprache modelliert [17, Abschnitt 2.1]. In einer solchen Formelmenge kann ein möglicher Satz (eine geschlossene Formel) aus der zugrunde liegenden logischen Sprache nicht enthalten sein, so dass sowohl Interpretationen, in denen dieser Satz gültig ist, als auch Interpretationen, in denen dieser Satz *nicht* gültig ist, die betrachtete Formelmenge erfüllen. Eine konkrete Modellierung einer unvollständigen Datenbankinstanz wird in Kapitel 5.2.1 diskutiert. Die dort vorgestellte Modellierung verfolgt dabei das (besondere) Ziel, trotz der Wahl einer unvollständigen Datenbankinstanz vollständiges Wissen zu repräsentieren.

Entsprechend dieser beiden Darstellungen logik-orientierter Datenbankinstanzen muss auch hinsichtlich der Semantik von Anfrageauswertungen unterschieden werden. Eine geschlossene Anfrage eines Benutzers an eine logik-orientierte Datenbankinstanz liegt dabei in Form einer (geschlossenen) Formel aus der logischen Sprache vor, mit der auch die logik-orientierte Datenbankinstanz dargestellt wird, an die die Anfrage gerichtet ist [9, Kap. 5.4]. Im Falle einer vollständigen Datenbankinstanz, die in der logik-orientierten Darstellung als Interpretation der verwendeten logischen Sprache aufgefasst werden kann (siehe oben), gilt eine solche Anfrage q genau dann als wahr, wenn q durch die Interpretation, die die betrachtete Datenbankinstanz beschreibt, erfüllt wird [12, Abschnitt 2.1]. Da eine Interpretation eine Formel entweder erfüllt oder nicht erfüllt (siehe oben), ist damit sichergestellt, dass die Anfrage q entweder eindeutig als wahr oder als nicht wahr in der betrachteten Datenbankinstanz klassifiziert werden kann.

Falls eine unvollständige Datenbankinstanz zum Einsatz kommt, wird hingegen mit Hilfe des Implikationsbegriffs der verwendeten logischen Sprache entschieden, welchen Wahrheitswert eine solche Anfrage q in der betrachteten Datenbankinstanz hat. Dabei gilt q als wahr in der mit db bezeichneten logik-orientierten Darstellung dieser Datenbankinstanz, falls q durch durch db impliziert wird. Falls hingegen $\neg q$ durch db impliziert wird, gilt q als nicht wahr in der betrachteten Datenbankinstanz. Falls weder q noch $\neg q$ durch db impliziert werden, gilt der Wahrheitswert für q als undefiniert [17, Abschnitt 2.1].

4.1.2 Vertraulichkeitspolitik eines Benutzers

Das erklärte Ziel der kontrollierten Anfrageauswertung ist es, den für einen Benutzer möglichen Informationsgewinn kontrolliert einzugrenzen. Dazu muss formal definiert werden können, welche konkreten Informations-Aspekte einem bestimmten Benutzer vorenthalten bleiben sollen. Diese explizite Formalisierung zu schützender Informations-Aspekte wird im Framework der kontrollierten Anfrageauswertung durch eine sogenannte Vertraulichkeitspolitik realisiert, die für jeden Benutzer des Systems definiert werden muss [10, Abschnitt 5].

Im einfachsten Fall besteht eine Vertraulichkeitspolitik aus einer Menge einzelner Informations-Aspekte, die als (geschlossene) Formeln der logischen Sprache, die auch der verwendeten logik-orientierten Datenbankinstanz zugrunde liegt (siehe Kapitel 4.1.1), formuliert werden. Dabei wird jede dieser Formeln als potentielles Geheimnis (potential secret) bezeichnet. Falls ein potentielles Geheimnis in der betrachteten Datenbankinstanz gültig ist, darf der Benutzer, für den dieses potentielle Geheimnis definiert ist, dies nicht erfahren können. Falls ein potentielles Geheimnis in der betrachteten Datenbankinstanz hingegen *nicht* gültig ist, darf der betreffende

Benutzer dies erfahren. Ein Benutzer muss es also aus rationalen Erwägungen heraus stets für möglich halten können, dass die potentiellen Geheimnisse aus der für ihn definierten Vertraulichkeitspolitik in der betrachteten Datenbankinstanz *nicht* gültig sind [10, Abschnitt 5].

Im Falle vollständiger Datenbankinstanzen kann auch eine Vertraulichkeitspolitik gegeben sein, die aus Paaren komplementärer Sätze aus der verwendeten logischen Sprache besteht. Dabei wird ein solches Paar $\{\Psi, \neg\Psi\}$, das jeweils eine geschlossene Formel Ψ und deren Negation $\neg\Psi$ enthält, auch als Heimlichkeit (secrecy) bezeichnet [12, Abschnitt 2.3]. Ziel einer solchen Heimlichkeit $\{\Psi, \neg\Psi\}$ ist es, dass ein Benutzer, in dessen Vertraulichkeitspolitik diese Heimlichkeit enthalten ist, nicht unterscheiden kann, ob in der betrachteten logik-orientierten Datenbankinstanz die durch Ψ oder die durch $\neg\Psi$ beschriebene Information gilt. Dem Benutzer ist lediglich bekannt, dass genau eine der beiden Möglichkeiten in der betrachteten Datenbankinstanz gelten muss. Diese Erkenntnis ergibt sich unmittelbar aus der Definition einer vollständigen Datenbankinstanz.

Bezüglich der Vertraulichkeitspolitik muss zudem festgelegt werden, ob einem Benutzer die für ihn definierte Vertraulichkeitspolitik bekannt ist oder ob diese vor ihm geheim gehalten wird [10, Abschnitt 5]. Falls die Vertraulichkeitspolitik einem Benutzer bekannt ist, muss bei der Ausgestaltung eines Verfahrens der kontrollierten Anfrageauswertung darauf geachtet werden, dass einem Benutzer die Kenntnis der für ihn definierten Vertraulichkeitspolitik nicht als Grundlage für Inferenzen dienen kann, die eben diese Vertraulichkeitspolitik verletzen (vgl. [12, Abschnitt 1]). In diesem Zusammenhang ist bezogen auf potentielle Geheimnisse anzumerken, dass eine Vertraulichkeitspolitik vollständig unabhängig von Wahrheitswerten aus der betrachteten Datenbankinstanz definiert wird. Die Existenz eines potentiellen Geheimnisses sagt also nichts über den Wahrheitswert dieses potentiellen Geheimnisses in der betrachteten Datenbankinstanz aus [17, Abschnitt 2.2].

Die Annahme, dass einem Benutzer die für ihn definierte Vertraulichkeitspolitik bekannt ist, hat den Vorteil, dass nicht auf die Geheimhaltung definierter Vertraulichkeitspolitiken geachtet werden muss. Falls einem Benutzer die für ihn definierte Vertraulichkeitspolitik hingegen unbekannt ist, können bei Verfahren der kontrollierten Anfrageauswertung in der Regel bessere Ergebnisse hinsichtlich der Verfügbarkeit von Information erzielt werden, als wenn von bekannten Vertraulichkeitspolitiken ausgegangen wird (vgl. [12, Abschnitt 4.1]). Dies liegt darin begründet, dass bei der Konstruktion von Verfahren der kontrollierten Anfrageauswertung im Falle einer unbekannten Vertraulichkeitspolitik nicht in Betracht gezogen werden muss, dass ein Benutzer das Wissen über die für ihn definierte Vertraulichkeitspolitik unter Umständen zum Erschließen von vertraulicher Information verwenden könnte. Dafür besteht die Gefahr, dass ein Benutzer vertrauliche Information erschließen

kann, falls er (unerwarteterweise) Kenntnis von der für ihn definierten Vertraulichkeitspolitik (oder Teilen dieser) erlangen sollte. Eine konkrete Ausgestaltung einer als bekannt angenommenen Vertraulichkeitspolitik in Form von potentiellen Geheimnissen wird in den Kapiteln 5.1.3 und 5.2.3 diskutiert.

4.1.3 Wissen eines Benutzers

Um im Rahmen von Verfahren der kontrollierten Anfrageauswertung beurteilen zu können, ob eine Anfrage eines Benutzers die für diesen Benutzer definierte Vertraulichkeitspolitik verletzen kann, müssen alle möglichen Inferenzen, die dieser Benutzer erschließen kann, ermittelt werden. Dazu ist es notwendig, das komplette Wissen, das einem Benutzer zur Verfügung steht, in dem jeweils verwendeten Framework der kontrollierten Anfrageauswertung zu modellieren.

Um bei dynamischen Verfahren der kontrollierten Anfrageauswertung (siehe Kapitel 4.2) das modellierte Wissen eines Benutzers stets auf dem aktuellen Stand zu halten, muss dieses Wissen um alle Informations-Aspekte, die ein Benutzer im Rahmen der Interaktion mit dem System (also beispielsweise durch Anfragen) hinzugewinnt, ergänzt werden. Dabei wird dieses Wissen in einer Menge von Formeln modelliert, die mit *log* bezeichnet wird [10, Abschnitt 4]. Im Falle einer betrachteten vollständigen Datenbankinstanz ist dabei zur Modellierung dieses Wissens die logische Sprache, durch die auch die betrachtete logik-orientierte Datenbankinstanz beschrieben wird, zweckmäßig (vgl. [12]).

Falls hingegen eine unvollständige Datenbankinstanz zum Einsatz kommt, ist es notwendig, im Wissen eines Benutzers ausdrücken zu können, dass der Wert einer Anfrage in der betrachteten Datenbankinstanz undefiniert sein kann (vgl. [17, Abschnitt 3.1]). Um diese Information durch Formeln in *log* modellieren zu können, muss die logische Sprache, durch die die betrachtete logik-orientierte Datenbankinstanz beschrieben wird, zur Modellierung des Wissens eines Benutzers zu einer modallogischen Sprache erweitert werden (siehe [17, Abschnitt 3.1]).

Bei Verfahren der kontrollierten Anfrageauswertung ist es unerlässlich, das komplette Wissen eines Benutzers zu modellieren (siehe oben). Deshalb muss auch das Vorwissen eines Benutzers (a priori knowledge) über Informations-Aspekte aus der betrachteten Datenbankinstanz und die Funktionsweise des verwendeten Systems in Form einer Menge *prior* von Formeln aus der verwendeten logischen Sprache erfasst werden [10, Abschnitte 4+5]. Zutreffende Annahmen zum Vorwissen eines Benutzers zu formulieren, ist zur verlässlichen Durchsetzung von Vertraulichkeitsanforderungen von entscheidender Bedeutung, weil auch dieses Vorwissen einem Benutzer zum Herleiten von Inferenzen zur Verfügung steht (vgl. [9, Kap. 4.1]).

Mögliche Ausgestaltungen des Vorwissens eines Benutzers werden im Rahmen dieser Diplomarbeit in den Kapiteln 5.2.5 und 5.2.6 behandelt.

4.2 Dynamischer Ansatz

Auf Basis der in Kapitel 4.1 vorgestellten Komponenten sollen nun die grundlegenden Ideen, auf denen die dynamischen Ansätze der kontrollierten Anfrageauswertung aufbauen, vorgestellt werden. Im Kontext der Inferenzkontrolle wird von dynamischen Ansätzen gesprochen, wenn der Informationsgewinn eines Benutzers zur Laufzeit des Systems kontrolliert und gegebenenfalls eingegrenzt wird [9, Kap. 4.3]. Das bedeutet, dass zur Laufzeit des Systems jede Antwort, die das System an einen Benutzer übermittelt, daraufhin überprüft werden muss, ob diese Antwort dem anfragenden Benutzer Inferenzen, die vertrauliche Information beinhalten, ermöglichen könnte. Wenn dies der Fall ist, muss das System vor Übermittlung der Antwort geeignet reagieren, um dem anfragenden Benutzer diesen Informationsgewinn vorenthalten zu können.

4.2.1 Ablauf der kontrollierten Anfrageauswertung

Der wesentliche Ablauf einer kontrollierten Anfrageauswertung ist schematisch in Abbildung 4.1 dargestellt (vgl. [9, Kap. 5.4]). Dabei ist diese Abbildung prinzipiell in zwei Bereiche zu unterteilen: Im linken Bereich dieser Abbildung sind mit der betrachteten logik-orientierten Datenbankinstanz, der für einen Benutzer definierten Vertraulichkeitspolitik und dem modellierten Wissen eines Benutzers die aus Kapitel 4.1 bekannten Komponenten dargestellt. Diese Komponenten kommen bei allen Verfahren der dynamischen kontrollierten Anfrageauswertung zum Einsatz. Im Folgenden wird daher davon ausgegangen, dass jede dieser Komponenten in einer bestimmten Ausprägung gegeben ist. Dabei muss darauf geachtet werden, dass die Kombination dieser gewählten Ausprägungen zusammen mit der gewünschten Reaktion des Systems auf potentiell gefährliche Antworten zulässig ist (siehe z.B. [10, Tabelle 2] oder für vollständige Datenbankinstanzen [12, Tabelle 1]).

Im rechten Bereich aus Abbildung 4.1 ist der Ablauf einer dynamischen kontrollierten Anfrageauswertung schematisch dargestellt. Eine solche Anfrageauswertung beginnt stets mit der Übermittlung der Anfrage q eines Benutzers an das System. Dabei hat q (wie aus Kapitel 4.1.1 bekannt) die Gestalt einer geschlossenen Formel aus der logischen Sprache, durch die auch die betrachtete logik-orientierte Datenbankinstanz beschrieben wird. Zuerst wird diese Anfrage q im Kontext der betrachteten logik-orientierten Datenbankinstanz ausgewertet. Dabei kann q entweder als

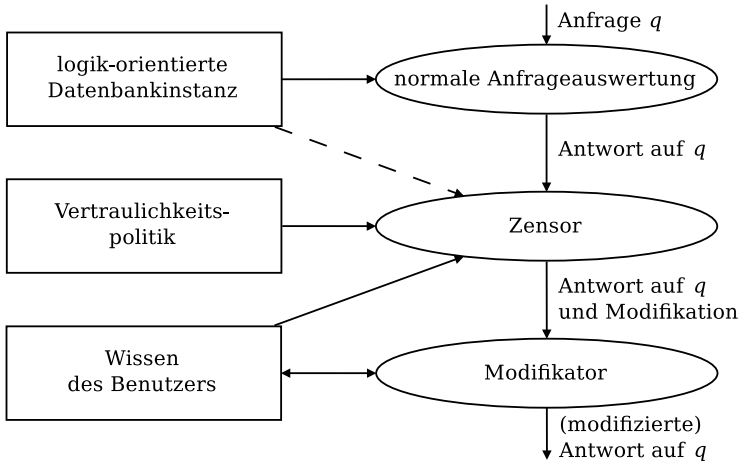

Abbildung 4.1: Ablauf der dynamischen kontrollierten Anfrageauswertung

wahr, nicht wahr oder im Falle einer betrachteten unvollständigen Datenbankinstanz auch als undefiniert klassifiziert werden (vgl. Kapitel 4.1.1).

Im Anschluss an die (normale) Auswertung von q muss überprüft werden, ob der anfragende Benutzer durch die Kenntnis des (korrekten) Wahrheitswertes für q unter Umständen Inferenzen herleiten kann, die die für diesen Benutzer definierte Vertraulichkeitspolitik verletzen würden [9, Kap. 5.4]. Dazu kommt ein Konzept zum Einsatz, das auch als Zensor bezeichnet wird. Im Rahmen dieses Konzepts wird überprüft, ob durch die Kombination aus

- dem Vorwissen des Benutzers,
- dem Wissen aus vorherigen Anfragen des Benutzers und
- dem Wissen über die Gültigkeit der Anfrage q

ein Informations-Aspekt impliziert wird, der diesem Benutzer gemäß der für ihn definierten Vertraulichkeitspolitik vorenthalten bleiben soll (vgl. [12, Abschnitte 2+3] und [17, Abschnitt 3.2]). Wenn dieser Fall eintritt, muss die korrekte Antwort auf Anfrage q durch das System der kontrollierten Anfrageauswertung derart modifiziert werden, dass die modifizierte Antwort keine solche Inferenzen erlaubt, die die Vertraulichkeitspolitik verletzen können.

Dies kann beispielsweise durch den in [36] entwickelten Ansatz des Verweigerns der Antworten auf derartige Anfragen erreicht werden. Unerwünschte Inferenzen durch das Verweigern von Antworten zu verhindern hat den Vorteil, dass ein Benutzer stets

davon ausgehen kann, dass alle nicht verweigerten Antworten des Systems auf seine Anfragen korrekt sind. Da aber bei Ansätzen der kontrollierten Anfrageauswertung davon ausgegangen wird, dass einem Benutzer die Funktionsweise des verwendeten Systems stets bekannt ist, ist unter der Annahme, dass ein Benutzer auch die für ihn definierte Vertraulichkeitspolitik kennt, Vorsicht geboten: In diesem Fall muss darauf geachtet werden, dass einem Benutzer aufgrund einer verweigerten Antwort keine unerwünschten Inferenzen ermöglicht werden [10, Abschnitt 6].

Um derartige Inferenzen zu verhindern, müssen bei Ansätzen der kontrollierten Anfrageauswertung, die Vertraulichkeit durch Verweigern von Antworten unter einer bekannten Vertraulichkeitspolitik erzielen sollen, zusätzlich auch Antworten verweigert werden, durch deren Kenntnis einem Benutzer eigentlich keine Verletzung der Vertraulichkeitspolitik ermöglicht wird. Dadurch soll ein Benutzer im Unklaren darüber bleiben, ob ihm eine bestimmte Antwort durch das System verweigert wird, weil ihm aufgrund der Kenntnis dieser Antwort tatsächlich unerwünschte Inferenzen ermöglicht würden, oder ob ihm eine Antwort verweigert wird, durch die ihm keine Verletzung der Vertraulichkeitspolitik möglich wäre [10, Abschnitt 6]. Durch diese Unkenntnis bezüglich des Grundes für die Verweigerung einer Antwort kann ein Benutzer die Tatsache, dass ihm eine Antwort verweigert wird, nicht zum Erschließen unerwünschter Information nutzen.

Alternativ zum Verweigern von Antworten können im Rahmen der kontrollierten Anfrageauswertung Vertraulichkeitsanforderungen auch durch den in [20] vorgestellten Ansatz durchgesetzt werden: In diesem wird vorgeschlagen, mögliche Verletzungen der Vertraulichkeitspolitik durch gezieltes Antworten mit falschen Antworten zu verhindern. Das heißt, dass die Antwort des Systems auf eine Anfrage q aus einem Wahrheitswert bestehen kann, der *nicht* dem Wahrheitswert von q in der betrachteten logik-orientierten Datenbankinstanz entspricht [10, Abschnitt 7]. Eine solche Antwort wird auch als Lüge bezeichnet. Im Gegensatz zu dem oben vorgestellten Ansatz des Verweigerns von Antworten muss bei diesem Ansatz die Sichtweise, die ein Benutzer auf die verwendete Datenbankinstanz aufgrund der Antworten des Systems auf seine Anfragen erhält, nicht zwangsläufig korrekt sein.

Dabei ist es für die Sicherheit dieses Ansatzes entscheidend, dass ein Benutzer trotz seiner Kenntnis, dass ihn das System eventuell belügt, nicht herausfinden kann, ob eine Antwort des Systems gelogen ist oder nicht. Dazu muss bei der Manipulation von Antworten durch das System darauf geachtet werden, dass ein Benutzer ausschließlich Antworten erhält, die konsistent zu seinem bisherigen Wissen sind. Insbesondere ist darauf zu achten, dass einem Benutzer keine Folge von Anfragen ermöglicht wird, die unter Verwendung des Lügen-Ansatzes zwangsläufig zu inkonsistentem Wissen des Benutzers führt. Es kann gezeigt werden, dass dieses Ziel erreicht wird, wenn ein Benutzer unter keinen Umständen erfahren kann, dass

eine beliebige Disjunktion von zu schützenden Informations-Aspekten aus seiner Vertraulichkeitspolitik gilt [10, Abschnitt 7].

Bei den oben vorgestellten Reaktionen des Systems ist auffällig, dass bei beiden Konzepten zur Durchsetzung von Vertraulichkeitsanforderungen (zumindest bei einer als bekannt angenommenen Vertraulichkeitspolitik) jeweils mehr Informations-Aspekte geschützt werden müssen als in der Vertraulichkeitspolitik definiert sind. Ein Ansatz, bei dem dies nicht notwendig ist, ist die in [13] vorgestellte kombinierte Methode, bei der sowohl Antworten verweigert als auch gelogene Antworten gegeben werden können. Bei dieser Methode müssen weder Antworten verweigert werden, die eigentlich keine Verletzung der definierten Vertraulichkeitspolitik ermöglichen würden, noch müssen bei dieser Methode mögliche Disjunktionen von zu schützenden Informations-Aspekten aus der Vertraulichkeitspolitik geschützt werden.

Nachdem durch den Zensor entschieden worden ist, ob die durch einen Benutzer gestellte Anfrage q abgelehnt oder (eventuell durch eine Lüge) beantwortet werden soll, muss unter Umständen noch das Wissen eines Benutzers – das durch das System der kontrollierten Anfrageauswertung für jeden Benutzer durch eine Menge *log* verwaltet wird – aktualisiert werden. Dabei muss die an den Benutzer übermittelte Antwort stets zu *log* hinzugefügt werden, falls dem Benutzer entweder eine korrekte oder eine gelogene Antwort übermittelt wird und die Information aus dieser Antwort noch nicht (direkt oder indirekt) in *log* enthalten ist. Die Information, dass eine Antwort auf eine bestimmte Anfrage verweigert wird, muss im Falle einer verwendeten vollständigen Datenbankinstanz nicht gespeichert werden (vgl. [12, Abschnitt 2.2]). Im Falle einer verwendeten unvollständigen Datenbankinstanz ist dies hingegen notwendig (vgl. [17, Abschnitt 3.1]).

4.2.2 Definition der Inferenzsicherheit

Aufgrund von Kapitel 4.2.1 ist nun der prinzipielle Ablauf eines Verfahrens der dynamischen kontrollierten Anfrageauswertung bekannt. Um die Inferenzsicherheit eines solchen Verfahrens beweisen zu können, kann für ein konkret betrachtetes Verfahren eine sogenannte deklarative Vertraulichkeitseigenschaft formuliert werden, deren Gültigkeit für dieses Verfahren formal nachzuweisen ist. Dabei muss darauf geachtet werden, dass die formulierte deklarative Vertraulichkeitseigenschaft geeignet auf das jeweils konkret betrachtete Verfahren abgestimmt ist (vgl. [10, Abschnitt 3]). Das heißt, dass sich insbesondere die Eigenschaften der gewählten Ausprägungen der einzelnen Komponenten (siehe Kapitel 4.1), die dem konkret betrachteten Verfahren der kontrollierten Anfrageauswertung zugrunde liegen, in einer geeigneten deklarativen Vertraulichkeitseigenschaft widerspiegeln müssen.

Im Folgenden soll basierend auf [10, Abschnitt 3], [12, Abschnitt 2.3] und [17, Abschnitt 2.3] eine allgemeine deklarative Vertraulichkeitseigenschaft vorgestellt werden, die für konkrete Verfahren der dynamischen kontrollierten Anfrageauswertung jeweils geeignet spezialisiert werden muss. Diese Vertraulichkeitseigenschaft besagt, dass ein Verfahren der dynamischen kontrollierten Anfrageauswertung genau dann als inferenzsicher bezüglich der Anfragen eines Benutzers gilt, wenn innerhalb der verwendeten logischen Sprache

- für *jede* mögliche (logik-orientierte) Datenbankinstanz,
- für *jede* mögliche Anfragesequenz dieses Benutzers,
- für *jedes* mögliche Vorwissen dieses Benutzers und
- für *jede* mögliche Vertraulichkeitspolitik dieses Benutzers

nachgewiesen werden kann, dass für *jedes* Element einer betrachteten Vertraulichkeitspolitik jeweils eine alternative (logik-orientierte) Datenbankinstanz konstruiert werden kann, für die die beiden folgenden Eigenschaften gelten:

(a) Die (kontrollierten) Antworten des Systems auf eine betrachtete Anfragesequenz sind unter der alternativen Datenbankinstanz ununterscheidbar zu den (kontrollierten) Antworten des Systems auf diese Anfragesequenz unter der eigentlichen Datenbankinstanz.

(b) Der Informations-Aspekt, der durch das betrachtete Element der Vertraulichkeitspolitik zu schützen ist, gilt in der alternativen Datenbankinstanz nicht.

Dabei muss darauf geachtet werden, dass die konstruierte alternative Datenbankinstanz gültig hinsichtlich der (logisch modellierten) semantischen Bedingungen ist, die auch für die eigentliche Datenbankinstanz zu erfüllen sind. Zudem muss für das Vorwissen eines Benutzers stets eine – für das jeweilige Verfahren der kontrollierten Anfrageauswertung geeignet formulierte – Vorbedingung erfüllt werden, damit der Nachweis der Inferenzsicherheit gelingt. Andernfalls könnte beispielsweise bereits in dem Vorwissen eines Benutzers ein geheim zu haltender Informations-Aspekt direkt enthalten sein (siehe Kapitel 5.2.5).

4.3 Statischer Ansatz

Bei der Entwicklung eines Verfahrens der dynamischen kontrollierten Anfrageauswertung ist gemäß Kapitel 4.2.2 Folgendes zu erreichen: Eine Sequenz von Anfragen eines Benutzers muss stets so beantwortet werden, dass dieser Benutzer aufgrund seines Vorwissens und der Kenntnis der (eventuell modifizierten) Antworten keine

Möglichkeit hat, durch rationale Erwägungen eine Datenbankinstanz zu erschließen, die die für ihn definierte Vertraulichkeitspolitik verletzt. Alternativ zu dieser Modifikation der Antworten zur Laufzeit des Systems kann auch vor Inbetriebnahme des Systems eine alternative Datenbankinstanz konstruiert werden, die einem Benutzer unter nicht kontrollierten Anfragesequenzen eine Verletzung der Vertraulichkeitspolitik gar nicht erst ermöglicht [10, Abschnitt 5]. Dieses Vorgehen wird auch als statische Inferenzkontrolle bezeichnet (vgl. [9, Kap. 4.3]).

Dazu werden in [18] und [19] geeignete Ansätze vorgestellt. In beiden Ansätzen wird davon ausgegangen, dass eine vollständige logik-orientierte Datenbankinstanz gegeben ist, für die eine durch potentielle Geheimnisse beschriebene Vertraulichkeitspolitik existiert, die dem betreffenden Benutzer bekannt ist. Ziel ist es jeweils, unter Einsatz möglicher Lügen eine inferenzsichere alternative Datenbankinstanz zu konstruieren, die möglichst wenig modifizierte Werte enthält. Zusätzlich kann eine explizite Verfügbarkeitspolitik angegeben werden, in der Informations-Aspekte aufgezählt werden, deren Wahrheitswerte sich – sofern dadurch nicht die Vertraulichkeitspolitik verletzt wird – in der alternativen Datenbankinstanz nicht von denen aus der eigentlichen Datenbankinstanz unterscheiden sollen [18, Abschnitt 3].

Damit eine alternative Datenbankinstanz unter beliebigen nicht kontrollierten Anfragen eines Benutzers die gewünschten Vertraulichkeitsanforderungen erfüllt, darf in dieser Datenbankinstanz keines der potentiellen Geheimnisse gültig sein [18, Abschnitt 3]. Des Weiteren muss darauf geachtet werden, dass die alternative Datenbankinstanz trotz dieser Modifikation in sich und zu dem Vorwissen des Benutzers konsistent bleibt, da ein Benutzer unter Einsatz der Lügen-Methode nicht erkennen können darf, dass er belogen wird (vgl. Kapitel 4.2.1).

Auch bei den statischen Ansätzen muss darauf geachtet werden, dass ein Benutzer aufgrund seines Vorwissens keine Möglichkeit hat, die Vertraulichkeitspolitik zu verletzen. Da ein Benutzer bei Einsatz der Lügen-Methode nicht erfahren können darf, dass eine beliebige Disjunktion zu schützender Informations-Aspekte gilt, darf das Vorwissen eines Benutzers keine solche Disjunktion enthalten [18, Abschnitt 2]. Diese Forderung wird automatisch auch für die zu konstruierende alternative Datenbankinstanz erfüllt: Wenn in dieser wie gefordert keines der potentiellen Geheimnisse aus der Vertraulichkeitspolitik gültig ist, kann in dieser auch keine Disjunktion von potentiellen Geheimnissen aus dieser Vertraulichkeitspolitik gültig sein.

5 Inferenzsicherheit der Fragmentierungs-Ansätze

In Kapitel 3 dieser Ausarbeitung werden verschiedene Verfahren zur fragmentierten Speicherung von Datenbankinstanzen vorgestellt. Ziel jedes dieser Verfahren ist es, basierend auf dem Konzept der Fragmentierung bestimmte Vertraulichkeitsanforderungen zu erfüllen, die in Form von Vertraulichkeits-Constraints formal ausgedrückt werden können. Dabei wird von den jeweiligen Autoren aber jeweils nur der Schutz vor unmittelbarem Zugriff auf vertrauliche Information diskutiert. Es wird nicht detailliert darauf eingegangen, ob die vorgestellten Verfahren die in den Vertraulichkeits-Constraints definierten Anforderungen auch unter Berücksichtigung eventuell möglicher Inferenzen (vgl. Kapitel 1.1.2) gewährleisten.

Im Gegensatz dazu ist für die in Kapitel 4 vorgestellten Verfahren zur kontrollierten Anfrageauswertung die Eigenschaft der Inferenzsicherheit formal nachgewiesen. Ziel dieses Kapitels ist es, die aus Kapitel 3 bekannten, auf Fragmentierung basierenden Verfahren im Framework der kontrollierten Anfrageauswertung zu formalisieren. Innerhalb dieses Frameworks – das Beweise zur Inferenzsicherheit der Verfahren der kontrollierten Anfrageauswertung ermöglicht – sollen auch beweisbare Aussagen zur Inferenzsicherheit der Verfahren zur fragmentierten Speicherung von Datenbankinstanzen getroffen werden können.

Dazu wird zuerst in Kapitel 5.1 ein geeignetes Framework der kontrollierten Anfrageauswertung entwickelt, in dem die in Kapitel 3 vorgestellten Fragmentierungs-Ansätze modelliert werden können. Im Anschluss daran wird in Kapitel 5.2 exemplarisch der Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ in dem entwickelten Framework modelliert. Innerhalb dieser Modellierung werden dann anschließend konkrete Untersuchungen hinsichtlich der Inferenzsicherheit dieses Fragmentierungs-Ansatzes durchgeführt.

5.1 Wahl des formalen Frameworks

Unter dem Oberbegriff „Kontrollierte Anfrageauswertung“ wird eine Vielzahl konkreter Verfahren zur kontrollierten Auswertung von Anfragen eines Benutzers zu-

sammengefasst. All diese Verfahren basieren (abstrahiert betrachtet) zwar im Wesentlichen auf den gleichen Klassen von Komponenten, allerdings unterscheiden sich die konkret gewählten Ausprägungen dieser Komponenten zwischen den verschiedenen konkreten Verfahren. Zudem existieren auch verschiedene Parameter, deren Wertebelegungen den Ablauf einer kontrollierten Anfrageauswertung entscheidend beeinflussen können.

Eine Übersicht über die bislang identifizierten Komponenten und deren konkrete Ausprägungen sowie über die bislang identifizierten Parameter und deren mögliche Wertebelegungen ist in [10, Abschnitt 5] zu finden. Basierend auf dieser Zusammenstellung sollen im Folgenden die Ausprägungen der wesentlichen Komponenten und die Werte der entscheidenden Parameter festgelegt werden, unter denen eine Modellierung der Fragmentierungs-Ansätze aus Kapitel 3 im Framework der kontrollierten Anfrageauswertung sinnvoll ermöglicht wird.

5.1.1 Wahl des logischen Systems

Eine Komponente, die bei Ansätzen zur kontrollierten Anfrageauswertung zwingend definiert werden muss, ist das verwendete logische System, in dem die betrachtete Datenbankinstanz, das Vorwissen eines Benutzers und die für einen Benutzer definierte Vertraulichkeitspolitik formalisiert werden sollen. Als logisches System soll hier in Anlehnung an [14, Abschnitt 2] eine eingeschränkte Prädikatenlogik 1. Ordnung mit Gleichheit zum Einsatz kommen. Diese logische Sprache wird sich als geeignet erweisen, um sowohl die aus Kapitel 3 bekannten Fragment-Instanzen, als auch die dort verwendete Vertraulichkeitspolitik in Form von Vertraulichkeits-Constraints im Kontext der kontrollierten Anfrageauswertung modellieren zu können. Um das verwendete logische System näher zu präzisieren, werden im Folgenden die wesentlichen Aspekte von dessen Syntax und Semantik erörtert. Dabei wird durch die Definition der Syntax eine formale Sprache $\mathscr{L}$ definiert, für die im Anschluss eine geeignete Semantik gefunden werden muss.

Zur Beschreibung der Syntax des verwendeten logischen Systems ist es zunächst notwendig, eine nicht leere, endliche Menge $\mathcal{P}$ von verwendbaren Prädikatensymbolen zu definieren. Dabei hat jedes Prädikatensymbol $P \in \mathcal{P}$ eine bestimmte Stelligkeit [19, Abschnitt 2]. Um eine relationale Datenbankinstanz zu einem relationalen Datenbankschema RS (vgl. Kapitel 2.1) in dem hier betrachteten logischen System modellieren zu können, ist es notwendig, dass für jedes in RS enthaltene Relationenschema $\langle R|A_R| \quad \rangle$ mit Relationensymbol R auch ein Prädikatensymbol R mit Stelligkeit $|A_R|$ in $\mathcal{P}$ enthalten ist.

Da es das erklärte Ziel ist, eine prädikatenlogische Sprache mit Gleichheit zu definieren, ist zusätzlich stets das ausgezeichnete Prädikatensymbol = mit Stelligkeit 2 implizit vorhanden, um Gleichheit auf Ebene der Syntax ausdrücken zu können [7, Kap. 6.1]. Dabei wird dieses ausgezeichnete Prädikatensymbol *nicht* als Element der Menge $\mathcal{P}$ angesehen, um es explizit getrennt von den Prädikatensymbolen aus $\mathcal{P}$, die zur Modellierung von Relationeninstanzen in der prädikatenlogischen Sprache verwendet werden, betrachten zu können.

Des Weiteren muss eine Domäne in Form einer Menge *Dom* von Konstantenzeichen vereinbart werden, die zur Konstruktion von Formeln (siehe unten) in der betrachteten prädikatenlogischen Sprache verwendet werden können. Dabei wird hier davon ausgegangen, dass die definierte Konstantenmenge stets unendlich groß und fest vereinbart ist [14, Abschnitt 1.7]. Die Annahme einer fest vereinbarten Konstantenmenge deckt sich dabei mit der Definition relationaler Datenbanken (vgl. Kapitel 2.1): Dort besteht eine Relationeninstanz einer relationalen Datenbankinstanz aus einer Menge von Tupeln und jedes dieser Tupel bildet die Attribute des zugrunde liegenden Relationenschemas jeweils auf einen Wert aus einer für das zugrunde liegende relationale Datenbankschema fest vereinbarten Konstantenmenge $\mathcal{K}$ ab. Im Unterschied zu dem relationalen Datenmodell wird bei der prädikatenlogischen Modellierung von Datenbanken allerdings auf die Typisierung der Attribute durch Zuordnung dieser zu bestimmten (attributspezifischen) Domänen verzichtet.

Die Einschränkung, dass die Menge *Dom* der Konstantenzeichen stets unendlich groß sein soll, ist dem Ziel des Nachweises der Inferenzsicherheit geschuldet, da bei unendlich großen Domänen Inferenzen durch sogenannte kombinatorische Effekte ausgeschlossen werden können. So könnte beispielsweise bei einer endlich großen Domäne in Form der Konstantenmenge $\{v_1, v_2\}$ und einer Menge gegebener Formeln (unten definiert) $\{(\exists X)(R(X)), \neg R(v_1)\}$ gefolgert werden, dass $R(v_2)$ gelten muss: Aufgrund der Formel $(\exists X)(R(X))$ muss mindestens ein Konstantenzeichen v in der Domäne existieren, für das $R(v)$ erfüllt ist. Da dies für v_1 wegen $\neg R(v_1)$ nicht der Fall ist, bleibt v_2 als einzig mögliche Alternative übrig [15, Abschnitt 3].

Auf Basis von Prädikatensymbolen können atomare prädikatenlogische Formeln definiert werden. Dabei kann für ein Prädikatensymbol $P \in \mathcal{P}$ mit Stelligkeit n eine atomare Formel $P(t_1, \ldots, t_n)$ konstruiert werden, wobei $t_1, \ldots, t_n$ sogenannte Terme sind [32, Kap. II.2]. Ein Term kann dabei entweder eine Konstante aus der Menge *Dom* aller Konstantenzeichen oder eine Variable aus der unendlich großen Menge $Var = \{X_1, X_2, \ldots\}$ aller Variablen sein. Im Gegensatz zu anderen gängigen Definitionen für die Syntax einer prädikatenlogischen Sprache (siehe z.B. [32, Kap. II.2]) können Terme hier nicht in Form von Funktionen, deren Funktionssymbole eine Stelligkeit echt größer als 0 haben[10], auftreten. Dies hat den Hintergrund, dass in

[10] Konstanten entsprechen Funktionssymbolen mit Stelligkeit 0 (siehe [32, Kap. II.2]).

gängigen Modellen für Relationeninstanzen Attributwerte nur fest (also konstant) definiert werden können. Funktionen, die eine dynamische Auswertung von Attributen erlauben würden, können hingegen nicht verwendet werden.

Auf Basis atomarer Formeln können mit Hilfe der Junktoren $\wedge$, $\vee$ und $\Rightarrow$ sowie der Negation $\neg$ komplexe Formeln über den gewöhnlichen induktiven Aufbau prädikatenlogischer Formeln (siehe z.B. [32, Kap. II.2]) konstruiert werden. Dabei können Variablen der Menge Var ausschließlich quantifiziert auftreten, so dass in der hier definierten prädikatenlogischen Sprache $\mathscr{L}$ ausschließlich geschlossene Formeln enthalten sein können. Zur Quantifizierung von Variablen stehen der Existenzquantor $\exists$ und der Allquantor $\forall$ zur Verfügung. Wie in Kapitel 5.2.3 zu sehen ist, kann es sich aber zu bestimmten Zwecken als notwendig erweisen, die Sprache $\mathscr{L}$ um freie Variablen zu erweitern. Mit Hilfe einer derart erweiterten Sprache können dann auch offene Anfragen an das System formuliert werden (siehe dazu [14]).

Für die oben beschriebene logische Sprache $\mathscr{L}$ in Form einer eingeschränkten Prädikatenlogik 1. Ordnung mit Gleichheit soll im Folgenden eine geeignete Semantik definiert werden. In dieser sollen sich – ebenso wie in der Syntax von $\mathscr{L}$ – die Eigenschaften relationaler Datenbanken widerspiegeln. Eine solche Semantik ist durch die aus [14, Abschnitt 2] und [19, Abschnitt 2] bekannte DB-Interpretation gegeben.

Definition 5.1 (DB-Interpretation) *Sei die prädikatenlogische Sprache $\mathscr{L}$ über einer festen, unendlich großen Domäne Dom und einer endlichen Menge $\mathcal{P}$ von Prädikatensymbolen gegeben. Eine prädikatenlogische Interpretation $\mathcal{I}$ über einem Universum $\mathcal{U}$ ist genau dann eine DB-Interpretation für $\mathscr{L}$, wenn*

(i) das Universum $\mathcal{U}$ gleich der Domäne Dom ist,

(ii) jedes Konstantenzeichen $v \in Dom$ über $\mathcal{I}(v) = v \in \mathcal{U}$ durch sich selbst interpretiert wird,

(iii) für jedes vorhandene Prädikatensymbol $P \in \mathcal{P}$ mit Stelligkeit n die zugehörige Interpretation $\mathcal{I}(P) \subset \underbrace{\mathcal{U} \times \ldots \times \mathcal{U}}_{n \text{ mal}}$ endliche Größe hat und

(iv) das ausgezeichnete Prädikatensymbol $= \notin \mathcal{P}$ stets durch die (unendlich große) binäre Relation $\mathcal{I}(=) = \{(v, v) \mid v \in \mathcal{U}\}$ interpretiert wird.

Eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$ ist also eine prädikatenlogische Interpretation, für die einige Einschränkungen gelten. Während das Universum $\mathcal{U}$, über dem eine Interpretation $\mathcal{I}$ definiert ist, im Falle einer gewöhnlichen Interpretation frei wählbar ist (siehe [32, Kap. II.4]), enthält das Universum im spezialisierten Fall einer DB-Interpretation genau die Elemente, die auch als Konstantenzeichen in der Domäne

Dom der (syntaktischen) Sprache $\mathscr{L}$ vorhanden sind. Da durch das Universum die (semantischen) Elemente aufgezählt werden, über denen eine Interpretation als Semantik für die zugrunde liegende syntaktische Sprache definiert werden kann, ist damit sichergestellt, dass eine beliebige DB-Interpretation für $\mathscr{L}$ immer genau über den Elementen definiert ist, die auch als Konstantenzeichen der (syntaktischen) Domäne von $\mathscr{L}$ zur Verfügung stehen.

Des Weiteren verbietet es Definition 5.1, dass ein Konstantenzeichen einer Formel aus $\mathscr{L}$ in einer DB-Interpretation durch ein beliebiges Element des Universums $\mathcal{U}$ interpretiert wird, was im Falle einer gewöhnlichen Interpretation möglich ist (siehe [32, Kap. II.4]). Bei einer DB-Interpretation, bei der das Universum $\mathcal{U}$ jedes Konstantenzeichen aus $\mathscr{L}$ auch als Element von $\mathcal{U}$ enthalten muss, muss ein Konstantenzeichen v aus der Domäne *Dom* der Sprache $\mathscr{L}$ stets durch das entsprechende Element $v \in \mathcal{U}$ des Universums interpretiert werden.

Da eine gewöhnliche Interpretation für ein Prädikatensymbol $P \in \mathcal{P}$ mit Stelligkeit n einer Relation in Form einer beliebigen Teilmenge von $\mathcal{U}^n$ entspricht (siehe [32, Kap. II.4]), kann eine solche Interpretation für P aufgrund der Annahme, dass $\mathcal{U}$ unendlich viele Elemente enthält, auch *unendlich* groß sein. Dies ist im Falle einer DB-Interpretation nicht möglich, da in Definition 5.1 explizit gefordert wird, dass eine DB-Interpretation für ein Prädikatensymbol aus $\mathcal{P}$ stets eine Relation in Form einer *endlich* großen Teilmenge von $\mathcal{U}^n$ sein muss. Ausgenommen von der Forderung nach endlicher Größe ist aber explizit das ausgezeichnete Prädikatensymbol $=$, für das auch eine ausgezeichnete feste Interpretation $\mathcal{I}(=) = \{(v, v) \mid v \in \mathcal{U}\}$ existiert [7, Kap. 6.1]. Um die Gleichheit zwischen allen Elementen des unendlich großen Universums $\mathcal{U}$ definieren zu können, ist offensichtlich auch eine unendlich große Relation notwendig, die für jedes $v \in \mathcal{U}$ genau ein Tupel $(v, v) \in \mathcal{I}(=)$ enthält.

Aufgrund dieser Einschränkungen kann eine DB-Interpretation $\mathcal{I}$ für eine konkrete prädikatenlogische Sprache $\mathscr{L}$ als eine vollständige relationale Datenbankinstanz aufgefasst werden, die durch $\mathcal{I}$ induziert wird. Die Sprache $\mathscr{L}$ ist zu einem relationalen Datenbankschema RS derart definiert, dass $\mathscr{L}$ für jedes in RS enthaltene Relationenschema $\langle R|A_R| \quad \rangle$ mit Relationensymbol R auch ein Prädikatensymbol R mit Stelligkeit $|A_R|$ in $\mathcal{P}$ enthält. Zudem wird die Konstantenmenge *Dom* von $\mathscr{L}$ so gewählt, dass $Dom := \mathcal{K}$ gilt, wenn mit $\mathcal{K}$ die für das relationale Datenbankschema RS vereinbarte (unendlich große) Konstantenmenge bezeichnet wird.

Für jedes Prädikatensymbol $P \in \mathcal{P}$ mit Stelligkeit n enthält eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$ eine Relation $\mathcal{I}(P) \subset \mathcal{U}^n$. Aufgrund dessen ist in $\mathcal{I}$ für ein Relationenschema $\langle R|A_R| \quad \rangle$ aus RS auch stets eine Menge $\mathcal{I}(R) \subset \mathcal{U}^{|A_R|}$ enthalten. Dabei repräsentiert in einem Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ jede Komponente v_i dieses Tupels einen Wert aus $\mathcal{U}$. Da in einer DB-Interpretation $\mathcal{U} = Dom$ gilt, gilt aufgrund der Wahl $Dom := \mathcal{K}$ also auch $\mathcal{U} = \mathcal{K}$. Da in einer DB-Interpretation des Weiteren

jedes Konstantenzeichen durch sich selbst interpretiert wird, entspricht ein Wert v_i aus $(v_1, \ldots, v_{|A_R|})$ auch genau dem Konstantenzeichen v_i aus $\mathcal{K}$. Damit kann zu Schema $\langle R|A_R| \quad \rangle$ eine Relationeninstanz r konstruiert werden, die für jedes Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ jeweils genau ein Tupel $\mu \in r$ enthält, in dem unter der Annahme $A_R = \{a_1, \ldots, a_{|A_R|}\}$ für alle $i \in \{1, \ldots, |A_R|\}$ jeweils $\mu[a_i] := v_i$ gilt. Da $\mathcal{I}(R)$ nach Definition einer DB-Interpretation endliche Größe hat, enthält damit auch r – wie in Definition 2.2 gefordert – endlich viele Tupel.

Auch hinsichtlich der Behandlung von negativem Wissen überdecken sich die beiden betrachteten Konzepte. Im Kontext einer vollständigen Relationeninstanz r wird alles Wissen, das nicht explizit aus Werten von Tupeln aus r als wahr hervorgeht, als nicht wahr in r angesehen (vgl. [12, Abschnitt 2.1]). Ebenso gilt im Sinne einer DB-Interpretation $\mathcal{I}$ alles Wissen, das nicht explizit in $\mathcal{I}$ enthalten ist, nicht: Wenn eine DB-Interpretation $\mathcal{I}$ (analog zu einer gewöhnlichen Interpretation) eine Formel Φ nicht erfüllt, erfüllt sie gemäß der Auswertung einer Formel über ihren induktiven Aufbau zwangsläufig deren Negation $\neg\Phi$ (siehe [32, Kap. II.4]).

Die Auswertung, ob eine DB-Interpretation $\mathcal{I}$ für eine konkrete Sprache $\mathscr{L}$ eine Formel Φ aus $\mathscr{L}$ erfüllt (kurz: $\mathcal{I} \models_M \Phi$), erfolgt dabei wie gewohnt anhand des induktiven Aufbaus der Formel Φ (siehe [32, Kap. II.4]). Eine Menge $\mathcal{S}$ von Formeln aus $\mathscr{L}$ wird genau dann durch eine DB-Interpretation $\mathcal{I}$ erfüllt (kurz: $\mathcal{I} \models_M \mathcal{S}$), wenn für jede Formel $\Phi \in \mathcal{S}$ jeweils $\mathcal{I} \models_M \Phi$ gilt. Falls eine Formel Φ eine existenzquantifizierte Variable X enthält, muss es mindestens *eine* mögliche Ersetzung dieser Variablen X in Φ durch ein beliebiges Konstantenzeichen $v \in Dom$ geben (notiert durch $\Phi(X/v)$), unter der die derart modifizierte Formel $\Phi(X/v)$ durch $\mathcal{I}$ erfüllt wird, damit auch die nicht modifizierte Formel Φ durch $\mathcal{I}$ erfüllt wird. Für den Fall, dass Φ eine allquantifizierte Variable X enthält, muss *jede* modifizierte Formel $\Phi(X/v)$, die durch Ersetzen von X durch ein beliebiges Konstantenzeichen $v \in Dom$ gebildet werden kann, durch $\mathcal{I}$ erfüllt werden, damit auch auch die nicht modifizierte Formel Φ durch $\mathcal{I}$ erfüllt wird.

Aufbauend auf dem Konzept der DB-Interpretation kann im Folgenden nach [14, Abschnitt 2] und [19, Abschnitt 2] ein passender Implikationsbegriff in Form der sogenannten DB-Implikation definiert werden.

Definition 5.2 (DB-Implikation) *Sei $\mathcal{S}$ eine Menge von Formeln aus der prädikatenlogischen Sprache $\mathscr{L}$ und sei Φ eine Formel aus $\mathscr{L}$. Die Formelmenge $\mathcal{S}$ impliziert die Formel Φ genau dann durch DB-Implikation (auch notiert durch $\mathcal{S} \models_{DB} \Phi$), wenn für jede gemäß Definition 5.1 gestaltete DB-Interpretation $\mathcal{I}$, für die $\mathcal{I} \models_M \mathcal{S}$ gilt, auch $\mathcal{I} \models_M \Phi$ gilt.*

Gemäß dieser Definition muss also die Menge aller DB-Interpretationen, die $\mathcal{S}$ erfüllen, eine Teilmenge der Menge aller DB-Interpretationen, die Φ erfüllen, sein, damit $\mathcal{S} \models_{DB} \Phi$ gilt.

5.1.2 Wahl des Typs der Datenbankinstanz

Bei den verschiedenen Ansätzen zur kontrollierten Anfrageauswertung wird grundsätzlich zwischen Ansätzen für vollständige und Ansätzen für unvollständige relationale Datenbankinstanzen unterschieden [10, Abschnitt 5]. Vollständige Datenbankinstanzen sind dadurch charakterisiert, dass sie auf Basis ihres Wissens jede mögliche geschlossene Anfrage klar als wahr oder als nicht wahr beantworten können. Deshalb kann eine solche Datenbankinstanz repräsentiert werden, indem alles positiv geltende Wissen als solches explizit modelliert wird. Alle geschlossenen Anfragen an das System, die aufgrund des vorhandenen positiven Wissens nicht explizit als wahr beantwortet werden können, werden als nicht wahr beantwortet (vgl. [12, Abschnitt 2.1]). Diese besondere Art der Behandlung von negativem Wissen ist auch unter dem Begriff „Closed World Assumption“ bekannt [32, Kap. III.6].

Bei (möglicherweise) unvollständigen Datenbankinstanzen muss hingegen aus dem vorhandenen Wissen explizit hergeleitet werden können, dass eine geschlossene Anfrage nicht wahr ist, damit diese auch als nicht wahr beantwortet wird [17, Abschnitt 2.1]. Deshalb kann es im Kontext unvollständiger Datenbankinstanzen geschlossene Anfragen geben, die weder explizit als wahr noch als nicht wahr beantwortet werden können. Solche Anfragen werden als undefiniert beantwortet, um zu signalisieren, dass das vorhandene Wissen in der betrachteten Datenbankinstanz nicht ausreichend ist, um die betreffende Anfrage eindeutig als wahr oder als nicht wahr klassifizieren zu können.

Im Rahmen der Fragmentierungs-Ansätze aus Kapitel 3 werden im Wesentlichen zwei verschiedene Arten von Relationeninstanzen betrachtet: Einerseits existiert eine ursprüngliche Relationeninstanz, die mit Hilfe von Projektionen in verschiedene Fragment-Instanzen zerlegt werden soll, und andererseits existieren eben diese Fragment-Instanzen, die ebenfalls Relationeninstanzen sind und im Prozess der Fragmentierung entstanden sind. Die ursprüngliche Relationeninstanz kann dabei als relationale Datenbankinstanz, die aus lediglich einer einzelnen Relationeninstanz besteht, aufgefasst werden. Im Folgenden dieser Ausarbeitung wird dabei unter der Annahme gearbeitet, dass diese Datenbankinstanz vollständig ist.[11]

[11] Die Autoren der entsprechenden Ausarbeitungen definieren für die dort betrachteten Relationeninstanzen nicht, ob diese vollständig oder unvollständig sind.

Jeweils für sich (aus der Sichtweise einer eigenständigen Relationeninstanz heraus) betrachtet gilt diese Eigenschaft der Vollständigkeit damit auch für alle Fragment-Instanzen, die im Prozess der Fragmentierung der ursprünglichen Relationeninstanz entstanden sind. Ziel dieser Ausarbeitung ist es aber, zu untersuchen, inwieweit Inferenzen hinsichtlich der ursprünglichen Relationeninstanz bei Kenntnis bestimmter Fragment-Instanzen möglich sind. Aus diesem Grund dürfen Fragment-Instanzen nicht aus der isolierten Sichtweise eigenständiger Relationeninstanzen heraus betrachtet werden, sondern müssen im Kontext der Fragmentierung als Teile der ursprünglichen Relationeninstanz behandelt werden, um den semantischen Bezug zu der ursprünglichen Relationeninstanz zu erhalten.

Im Rahmen dieser Sichtweise auf Fragment-Instanzen müssen die Ideen zur Konstruktion einer Fragment-Instanz auf Basis der ursprünglichen Relationeninstanz mit einbezogen werden, um eine korrekte Modellierung dieser Sichtweise zu ermöglichen. Dabei ist entscheidend, dass jedes Fragment gemäß der Definition der allgemeinen Fragmentierung (siehe Definition 3.1) – auf der alle spezielleren Definitionen der konkreten Fragmentierungs-Ansätze aus Kapitel 3 aufbauen – eine Teilmenge der Attribute des Schemas der ursprünglichen Relationeninstanz enthält. Des Weiteren ist eine Fragment-Instanz zu einem solchen Fragment eine Projektion der ursprünglichen Relationeninstanz auf die Teilmenge der Attribute, die in dem betrachteten Fragment enthalten ist.

Das heißt, dass jedes Tupel μ aus einer ursprünglichen Relationeninstanz r prinzipiell auch in einer beliebigen Fragment-Instanz f, die zu r gebildet wird, enthalten ist. Allerdings ist ein solches Tupel μ in f nicht komplett vorhanden, sondern auf die Attributmenge des Fragments F eingeschränkt, zu dem die Fragment-Instanz f bezüglich r gebildet wurde. Das führt dazu, dass man grundsätzlich zwischen zwei Teilmengen von Attributen des r zugrunde liegenden Schemas R unterscheiden muss: Einerseits existiert die Teilmenge der Attribute, die sowohl in R als auch in Fragment F enthalten sind. Für jedes dieser Attribute sind alle in r existierenden Attributwerte und die Assoziationen zwischen diesen auch in f enthalten. Andererseits existiert die Teilmenge der Attribute, die ausschließlich in R und nicht in F enthalten sind. Für diese Attribute ist keiner der zugehörigen Attributwerte aus r in f enthalten.

Man kann f im Kontext der ursprünglichen Relationeninstanz r also als eingeschränkte Sichtweise von r interpretieren. Diese Sichtweise der ursprünglichen Relationeninstanz kann selbst wiederum als Datenbankinstanz aufgefasst werden, die geschlossene Anfragen bezüglich der in F enthaltenen Attribute stets korrekt als wahr oder nicht wahr beantworten kann. Anfragen bezüglich der Attribute von Schema R, die nicht in F enthalten sind, können von dieser Datenbankinstanz hingegen weder eindeutig als wahr, noch eindeutig als nicht wahr beantwortet werden,

R	a_1	a_2	a_3
	$\mu_1[a_1]$	$\mu_1[a_2]$	$\mu_1[a_3]$
	$\mu_2[a_1]$	$\mu_2[a_2]$	$\mu_2[a_3]$
	$\mu_3[a_1]$	$\mu_3[a_2]$	$\mu_3[a_3]$

(a) Ursprüngliche Instanz r

F	a_1	a_2
	$\mu_1[a_1]$	$\mu_1[a_2]$
	$\mu_2[a_1]$	$\mu_2[a_2]$
	$\mu_3[a_1]$	$\mu_3[a_2]$

(b) Fragment-Instanz f

$R(F)$	a_1	a_2	a_3
	$\mu_1[a_1]$	$\mu_1[a_2]$	?
	$\mu_2[a_1]$	$\mu_2[a_2]$	?
	$\mu_3[a_1]$	$\mu_3[a_2]$	?

(c) f im Kontext von r

Abbildung 5.1: Fragment im Kontext seiner ursprünglichen Relationeninstanz

da innerhalb dieser Sichtweise das notwendige Wissen dazu fehlt. Aus diesem Grund muss diese Sichtweise von Fragment-Instanzen im Kontext ihrer ursprünglichen Relationeninstanz als unvollständige Datenbankinstanz modelliert werden, obwohl die ursprüngliche Relationeninstanz als vollständig definiert ist.

Eine solche Modellierung ist in einem logischen Framework auf jeden Fall möglich, da laut [8, Abschnitt 3.1] die vollständige Darstellung einer vollständigen Datenbankinstanz stets in eine unvollständige Darstellung dieser Datenbankinstanz überführt werden kann, ohne dass dabei das implizit vorhandene negative Wissen, das in der ursprünglichen Darstellung durch die Eigenschaft der Vollständigkeit vorhanden ist, verloren geht. In der unvollständigen Darstellung einer betrachteten ursprünglichen, vollständigen Relationeninstanz kann dann das Wissen dieser ursprünglichen Instanz entfernt werden, das nicht in den betrachteten Fragment-Instanzen enthalten ist und deshalb auch in der zu modellierenden Sichtweise der ursprünglichen Relationeninstanz nicht vorhanden sein soll.

Diese Beziehung zwischen einer gegebenen ursprünglichen Relationeninstanz und den Fragment-Instanzen, die im Zuge der Fragmentierung zu dieser Relationeninstanz gebildet wurden, soll beispielhaft in Abbildung 5.1 verdeutlicht werden. In Abbildungsteil 5.1(a) ist dabei eine mögliche ursprüngliche Relationeninstanz r über einem Relationenschema R zu sehen, das die Attributmenge $\{a_1, a_2, a_3\}$ enthält. Dabei besteht r aus den Tupeln μ_1, μ_2 und μ_3 und jedes dieser Tupel ordnet jedem der drei Attribute einen bestimmten Wert zu. Eine mögliche Fragment-Instanz bezüglich r ist mit f in Abbildungsteil 5.1(b) gegeben. Dabei ist f zu einem Schema F gebildet, welches die Teilmenge $\{a_1, a_2\}$ der Attributmenge von R enthält.

Entsprechend der Definition einer Fragment-Instanz entspricht jedes Tupel aus f mindestens einem Tupel aus der ursprünglichen Relationeninstanz r, das auf die Attributmenge $\{a_1, a_2\}$ eingeschränkt ist.

Wenn man diese Fragment-Instanz f im Kontext ihrer ursprünglichen Relationeninstanz r betrachtet, erhält man die unvollständige Sichtweise, die in Abbildungsteil 5.1(c) dargestellt ist. Dabei können keine geschlossenen Anfragen, die einen konkreten Attributwert zu Attribut a_3 betreffen, als wahr oder nicht wahr beantwortet werden, da das dazu notwendige Wissen in Form der Attributwerte zu a_3 in der Datenbankinstanz, die sich aus dieser Sichtweise heraus ergibt, nicht enthalten ist. Wie oben bereits argumentiert, ist es aus diesem Grund notwendig, diese Sichtweise als unvollständige Datenbankinstanz zu modellieren.[12]

5.1.3 Wahl des Typs der Vertraulichkeitspolitik

Eine weitere entscheidende Komponente, die es bei jedem konkreten Framework der kontrollierten Anfrageauswertung zu spezifizieren gilt, ist der Typ der verwendeten Vertraulichkeitspolitik. In dieser wird explizit definiert, welche Informations-Aspekte einem bestimmten Benutzer (oder auch einer Klasse von Benutzern) vorenthalten bleiben sollen. Dazu existieren mit den sogenannten potentiellen Geheimnissen (potential secrets) und den sogenannten Heimlichkeiten (secrecies) zwei wesentliche Typen von Vertraulichkeitspolitiken [10, Abschnitt 5]. Ziel muss es also sein, einen Typ Vertraulichkeitspolitik zu identifizieren, der dazu geeignet ist, das Konzept der Vertraulichkeits-Constraints – das im Rahmen der Fragmentierungs-Ansätze zur Definition von Vertraulichkeitsanforderungen eingesetzt wird (vgl. Kapitel 3.1.2) – im Framework der kontrollierten Anfrageauswertung auszudrücken.

Dazu sollen im Folgenden noch einmal kurz die wesentlichen Aspekte des Konzepts der Vertraulichkeits-Constraints wiederholt werden, die als Entscheidungsgrundlage für die Wahl des gewünschten Typs der Vertraulichkeitspolitik von Relevanz sind. Ein Vertraulichkeits-Constraint c ist durch eine Teilmenge der Attribute eines Relationenschemas gegeben. Im Falle eines einfachen Vertraulichkeits-Constraints $c = \{a\}$ sollen einem nicht autorisierten Betrachter die Werte, die dem Attribut a dieses Schemas in einer Relationeninstanz zu diesem Schema zugeordnet werden, vorenthalten bleiben.

Im Falle eines assoziierenden Vertraulichkeits-Constraints $c = \{a_1, \ldots, a_k\}$ (es gilt $|c| > 1$) darf ein nicht autorisierter Betrachter hingegen Kenntnis von den Wer-

[12] Im Rahmen der Modellierung dieser Sichtweise als unvollständige Datenbankinstanz muss das in der (vollständigen) Fragment-Instanz f implizit vorhandene negative Wissen explizit modelliert werden. Dies soll an dieser Stelle aber nicht weiter betrachtet werden.

ten, die einzelnen Attributen aus c in der betrachteten Relationeninstanz zu diesem Schema zugeordnet werden, erlangen. Ebenso dürfen ihm Wertekombinationen bekannt sein, die einer *echten* Teilmenge der Attribute aus c in Tupeln einer solchen Relationeninstanz zugeordnet werden. Er darf aber keine Kenntnis von den Wertekombinationen, die den Attributen $a_1, \ldots, a_k$ jeweils in den einzelnen Tupeln einer solchen Relationeninstanz zugeordnet werden, erlangen. Es sind im Falle assoziierender Vertraulichkeits-Constraints also Assoziationen zwischen verschiedenen Informations-Aspekten in Form von Attributwerten zu schützen.

Dabei wird hier unter der Annahme gearbeitet, dass sich die durch Vertraulichkeits-Constraints definierten Schutzanforderungen nur auf Wertekombinationen (im Fall von einfachen Vertraulichkeits-Constraints kann von zu schützenden einelementigen Wertekombinationen gesprochen werden) beziehen, die auch tatsächlich in Tupeln der betrachteten Relationeninstanz vorkommen. Für die möglichen Wertekombinationen der Attribute eines Vertraulichkeits-Constraints, die zwar aufgrund der Domänen der jeweiligen Attribute denkbar wären, aber konkret in keinem Tupel der betrachteten Relationeninstanz vorkommen, werden durch Vertraulichkeits-Constraints unter dieser getroffen Annahme keine Schutzanforderungen definiert. Ein nicht autorisierter Betrachter darf aufgrund dieser Annahme also beispielsweise in Erfahrung bringen können, dass eine bestimmte – prinzipiell durch ein Vertraulichkeits-Constraint geschützte – Wertekombination in *keinem* Tupel der betrachteten Relationeninstanz enthalten ist.

Diese Annahme muss hier getroffen werden, um die Semantik von Vertraulichkeits-Constraints für die Wahl eines geeigneten Frameworks der kontrollierten Anfrageauswertung genau genug zu spezifizieren. In einem Framework der kontrollierten Anfrageauswertung ist bezogen auf eine zum Einsatz kommende Vertraulichkeitspolitik genau zu definieren, welche konkreten Informations-Aspekte aus einer Datenbankinstanz einem Benutzer aufgrund dieser Vertraulichkeitspolitik vorenthalten werden sollen. Zu diesem Zweck existieren für Vertraulichkeitspolitiken der kontrollierten Anfrageauswertung formalisierte Semantiken, in denen klar zwischen der Schema- und der Instanz-Ebene einer Datenbankrelation differenziert wird (siehe z.B. [12, Abschnitt 2.3]).

Im Gegensatz dazu wird die Semantik von Vertraulichkeits-Constraints im Rahmen der Fragmentierungs-Ansätze ausschließlich auf Ebene des Schemas formal definiert (siehe z.B. [27, Definition 3.3]). Für die Semantik von Vertraulichkeits-Constraints werden bezogen auf Instanzen zu einem Schema hingegen keine formalen Aussagen getroffen. Deshalb wird in den entsprechenden Ausarbeitungen nicht genau definiert, ob einem nicht autorisierten Betrachter auch die Information, dass eine Wertekombination für die Attribute eines Vertraulichkeits-Constraints – die aufgrund des Schemas möglich ist – in *keinem* Tupel einer betrachteten Relationenin-

stanz vorkommt, vorenthalten bleiben muss. Als Folge dessen wird hier unter der oben genannten Annahme gearbeitet, durch die die Semantik von Vertraulichkeits-Constraints bezogen auf Relationeninstanzen konkretisiert wird.

Unter der oben formulierten Annahme ist zum Ausdruck derartiger Vertraulichkeitsanforderungen im Framework der kontrollierten Anfrageauswertung eine Vertraulichkeitspolitik in Form einer Menge pot_sec von potentiellen Geheimnissen geeignet. Bei einem potentiellen Geheimnis $\Psi \in pot\_sec$ handelt es sich um eine geschlossene Formel (auch Satz genannt) aus der logischen Sprache, die dem jeweils konkret betrachteten Framework der kontrollierten Anfrageauswertung zugrunde liegt. In diesem Fall handelt es sich dabei also um die aus Kapitel 5.1.1 bekannte prädikatenlogische Sprache $\mathscr{L}$.

Ein potentielles Geheimnis Ψ wird für einen bestimmten Benutzer mit dem Ziel definiert, dass es aus Sicht dieses Benutzers stets als möglich erscheinen soll, dass der durch Ψ beschriebene Informations-Aspekt in der betrachteten Datenbankinstanz – unabhängig von dessen wirklichem Wahrheitswert – nicht gilt [17, Abschnitt 2.2]. Das bedeutet, dass dieser Benutzer für ein in der betrachteten Datenbankinstanz gültiges potentielles Geheimnis Ψ – auch unter Ausnutzung eventuell möglicher Inferenzen – *nicht* erfahren können darf, dass Ψ gilt. Falls Ψ in der betrachteten Datenbankinstanz hingegen nicht gilt, darf der Benutzer dies erfahren. In diesem Fall gilt entweder $\neg\Psi$, oder im Falle einer betrachteten unvollständigen Datenbankinstanz kann der Wahrheitswert für Ψ auch undefiniert sein.

Offensichtlich muss sowohl im Fall von Vertraulichkeits-Constraints als auch im Fall von potentiellen Geheimnissen verhindert werden, dass ein dazu nicht autorisierter Benutzer in Erfahrung bringen kann, dass ein Element, welches gemäß der jeweiligen Vertraulichkeitspolitik zu schützen ist, in der jeweils betrachteten Datenbankinstanz gültig ist. Aufgrund dieser Übereinstimmung der beiden Semantiken bietet es sich an, Vertraulichkeits-Constraints im Framework der kontrollierten Anfrageauswertung in Form von potentiellen Geheimnissen zu modellieren.

Dabei muss beachtet werden, dass Vertraulichkeits-Constraints ausschließlich durch Teilmengen der Attribute eines Datenbankschemas definiert werden. Deshalb werden durch ein Vertraulichkeits-Constraint unter der oben getroffenen Annahme bezogen auf eine konkrete Relationeninstanz stets Schutzanforderungen für *alle* Informations-Aspekte, die in dieser Relationeninstanz durch die Wertekombinationen für die Attribute dieses Vertraulichkeits-Constraints beschrieben werden, zum Ausdruck gebracht. Potentielle Geheimnisse werden hingegen durch Formeln der logischen Sprache beschrieben, die dem verwendeten Framework der kontrollierten Anfrageauswertung zugrunde liegt und in deren Domäne alle Konstantenzeichen enthalten sind, die auch für das Datenbankschema der betrachteten Datenbankinstanz vereinbart sind. Damit können Formeln aus dieser logischen Sprache insbe-

sondere auch Konstantenzeichen aus der Domäne dieser Sprache enthalten. Als Folge dessen können durch potentielle Geheimnisse bei Bedarf auch direkt Schutzanforderungen für möglicherweise in einer Datenbankinstanz enthaltene einzelne Informations-Aspekte beschrieben werden.

5.1.4 Annahmen bezüglich nicht autorisierter Betrachter

In Kapitel 3 wird im Rahmen der dort vorgestellten Fragmentierungs-Ansätze prinzipiell zwischen zwei Klassen von Benutzern unterschieden: Einerseits wird die Existenz autorisierter Benutzer vorausgesetzt, die Zugriff auf den kompletten Datenbestand einer ursprünglichen Relationeninstanz haben sollen. Deshalb müssen diese die Möglichkeit haben, alle zu der ursprünglichen Relationeninstanz entstandenen Fragment-Instanzen wieder korrekt im Sinne der ursprünglichen Relationeninstanz zusammensetzen zu können. Andererseits wird von der Existenz nicht autorisierter Betrachter ausgegangen. Diese können unter Umständen zwar Zugriff auf eine bestimmte Teilmenge der entstandenen Fragment-Instanzen bekommen, sollen auf Basis dieses Wissens aber unter keinen Umständen die Möglichkeit haben, Kenntnis von Informations-Aspekten aus der ursprünglichen Relationeninstanz zu erlangen, die durch Vertraulichkeits-Constraints als schützenswert deklariert sind.

Nicht autorisierte Betrachter können im Kontext von Untersuchungen zur Inferenzsicherheit der Fragmentierungs-Ansätze also als potentielle Angreifer angesehen werden, die auf Basis von (unerwünschten) Inferenzen unter Umständen Information erschließen möchten, die ihnen eigentlich vorenthalten bleiben sollte. Im Vorfeld dieser Untersuchungen müssen deshalb geeignete Annahmen bezüglich des Wissens und der Fähigkeiten eines solchen Betrachters getroffen werden, um im Rahmen der Untersuchungen zur Inferenzsicherheit Kenntnisse über dessen Möglichkeiten zum Schließen von Inferenzen zu haben (vgl. [10, Abschnitt 5]).

Ziel soll es dabei grundsätzlich sein, einem nicht autorisierten Betrachter möglichst weitreichendes Wissen über das betrachtete System zu erlauben und diesem möglichst weitreichende Fähigkeiten hinsichtlich des (rationalen) Erschließens von Information zuzugestehen. Diese Grundsätze sind durch die Idee motiviert, dass die Sicherheit eines Systems wenn möglich nicht auf der Annahme beruhen sollte, dass einem potentiellen Angreifer Wissen über die Funktionsweise einzelner Komponenten des Systems vorenthalten bleibt (vgl. [4, Kap. 13.8]). Ebenso sollte (ohne gute, formal nachgewiesene Gründe) auch nicht darauf vertraut werden, dass diesem die notwendigen Fähigkeiten zum Berechnen von Inferenzen, die vertrauliche Information beinhalten, fehlen. Damit gelten Eigenschaften bezüglich der Inferenzsicherheit

auch unter Annahmen, die das Wissen und die Fähigkeiten eines Angreifers restriktiver einschätzen, wenn diese Eigenschaften für die hier vorausgesetzten weniger restriktiven Annahmen nachgewiesen sind.

Bezogen auf das Wissen eines nicht autorisierten Betrachters wird im Folgenden davon ausgegangen, dass dieser Kenntnis über das Relationenschema $\langle R|A_R|SC_R\rangle$, das der ursprünglichen Relationeninstanz r zugrunde liegt, hat. Ebenso ist ihm die Menge $\mathcal{C}$ der für das Schema $\langle R|A_R|SC_R\rangle$ definierten Vertraulichkeits-Constraints sowie deren Semantik (siehe Kapitel 5.1.3) bekannt. Des Weiteren kennt ein nicht autorisierter Betrachter auch die in Kapitel 3 vorgestellten Verfahren zur Fragmentierung sowie die konkreten Algorithmen, die jeweils zur Berechnung einer Fragmentierung $\mathcal{F}$ auf Basis eines gegebenen Relationenschemas und einer gegebenen Menge von Vertraulichkeits-Constraints eingesetzt wurden. Aus der Kenntnis des ursprünglichen Relationenschemas, der Menge der Vertraulichkeits-Constraints und des konkret verwendeten Algorithmus folgt, dass einem nicht autorisierten Betrachter auch die entstandene Fragmentierung $\mathcal{F}$, auf deren Basis die Fragment-Instanzen zu r erzeugt wurden, nicht vorenthalten werden kann.

Nicht bekannt sind einem nicht autorisierten Betrachter insbesondere die ursprüngliche Relationeninstanz r und die Fragment-Instanzen, die ihm gemäß des jeweils konkret verwendeten Fragmentierungs-Ansatzes explizit vorenthalten bleiben sollen. Ebenso sind ihm eventuell verwendete kryptographische Schlüssel unbekannt. Von diesen Einschränkungen hinsichtlich des Wissens eines nicht autorisierten Betrachters kann nicht abgesehen werden, da diese als Voraussetzungen für die Anwendbarkeit des jeweiligen Fragmentierungs-Ansatzes vorgegeben sind und deren Missachtung unmittelbar zu einer Verletzung der definierten Vertraulichkeitsanforderungen führen würde.

Hinsichtlich der Fähigkeiten eines nicht autorisierten Betrachters wird im Folgenden davon ausgegangen, dass dieser ein sogenannter perfekter Angreifer ist [10, Abschnitt 5]. Das bedeutet insbesondere, dass alle Schlussfolgerungen dieses Betrachters ausnahmslos durch rationale Erwägungen fundiert sind und deshalb innerhalb des verwendeten Frameworks stets korrekt sind. Diese Art von Inferenzen wird gemäß der Charakterisierung nach Peirce auch als Deduktion bezeichnet [5, Kap. 3.1.3]. Des Weiteren wird ein perfekter Angreifer als omnipotent hinsichtlich seiner Möglichkeiten zur Berechnung von Inferenzen angesehen. Das bedeutet konkret, dass – jeweils unabhängig davon, ob dies aus Sicht der Komplexitätstheorie lösbar oder aus Sicht der Berechenbarkeitstheorie möglich erscheint – stets davon ausgegangen werden muss, dass dieser Angreifer alle Inferenzen, die innerhalb des verwendeten Frameworks theoretisch durch Deduktion hergeleitet werden können, auch praktisch berechnen kann [9, Kap. 4.1].

5.2 Inferenzsicherheit bei partieller lokaler Verwaltung

Aufbauend auf den grundlegenden Überlegungen aus Kapitel 5.1 zur Wahl eines geeigneten formalen Frameworks der kontrollierten Anfrageauswertung soll im Folgenden der in Kapitel 3.4 vorgestellte Ansatz „Fragmentierung und partielle lokale Verwaltung“ hinsichtlich seiner Inferenzsicherheit untersucht werden. Dazu müssen zuerst Überlegungen getätigt werden, wie dieser Ansatz konkret in dem gewählten Framework der kontrollierten Anfrageauswertung modelliert werden kann. Innerhalb dieser Modellierung sollen dann anschließend Sicherheitseigenschaften dieses Fragmentierungs-Ansatzes analysiert werden.

5.2.1 Modellierung der ursprünglichen Relationeninstanz

Um mögliche Inferenzen, die sich auf Basis der Kenntnis von Fragment-Instanzen ergeben, analysieren zu können, ist es nach den Überlegungen aus Kapitel 5.1.2 sinnvoll, Fragment-Instanzen im Kontext des Prinzips der Fragmentierung als Teile ihrer ursprünglichen Relationeninstanz zu betrachten. Um diesen semantischen Bezug auch in einer formalen Modellierung berücksichtigen zu können, sollen Fragment-Instanzen in Kapitel 5.2.2 in demselben logischen Framework modelliert werden, in dem auch deren ursprüngliche Relationeninstanz logisch repräsentiert werden kann. Aus diesem Grund erscheint es sinnvoll, im Vorfeld der Erläuterungen zur logikorientierten Modellierung von Fragment-Instanzen erst einmal eine konkrete logikorientierte Modellierung einer ursprünglichen Relationeninstanz, die in Fragment-Instanzen zerlegt werden kann, vorzustellen.

Um die prädikatenlogische Modellierung von Fragment-Instanzen derart zu gestalten, dass diese im Framework ihrer ursprünglichen Relationeninstanz dargestellt werden können, hat es sich in Kapitel 5.1.2 als zweckmäßig herausgestellt, für die prädikatenlogische Modellierung den Ansatz einer möglicherweise unvollständigen Datenbankinstanz zu wählen. Eine solche unvollständige Datenbankinstanz kann in Anlehnung an [17, Abschnitt 2.1] folgendermaßen definiert werden:

Definition 5.3 (Unvollständige Datenbankinstanz) *Sei die aus Kapitel 5.1.1 bekannte prädikatenlogische Sprache $\mathscr{L}$ gegeben. Eine (möglicherweise) unvollständige Datenbankinstanz über $\mathscr{L}$ ist durch eine endliche, konsistente Menge von Formeln aus $\mathscr{L}$ gegeben.*

Dabei wird eine Menge $\mathcal{S}$ von Formeln aus $\mathscr{L}$ genau dann als konsistent bezeichnet, wenn es mindestens eine DB-Interpretation $\mathcal{I}$ gibt, für die $\mathcal{I} \models_M \mathcal{S}$ gilt, so dass alle Formeln $\Phi \in \mathcal{S}$ durch dieselbe DB-Interpretation $\mathcal{I}$ erfüllt werden können [11,

Abschnitt 2.1]. Wenn $\mathscr{L}$ beispielsweise das einstellige Prädikatensymbol P und das Konstantenzeichen v enthält, wird durch die endliche und (offensichtlich) konsistente Formelmenge $\{P(v)\}$ nach Definition 5.3 eine möglicherweise unvollständige Datenbankinstanz beschrieben. Die Formelmenge $\{P(v), \neg P(v)\}$ beschreibt hingegen keine gültige Datenbankinstanz.

In Kapitel 3 wird bei allen vorgestellten Ansätzen davon ausgegangen, dass eine einzelne Relationeninstanz r, der ein Relationenschema $\langle R|A_R| \quad \rangle$ zugrunde liegt, fragmentiert werden soll. Diese Relationeninstanz kann dabei als eine relationale Datenbankinstanz *db* aufgefasst werden, die aus lediglich einer einzelnen Relationeninstanz r besteht und über einem relationalen Datenbankschema RS definiert ist, das lediglich das Relationenschema $\langle R|A_R| \quad \rangle$ enthält. Aus diesem Grund enthält die in Kapitel 5.1.1 eingeführte prädikatenlogische Sprache $\mathscr{L}$ zur logik-orientierten Modellierung der ursprünglichen Relationeninstanz r hier nur ein einziges Prädikatensymbol R in der Menge $\mathcal{P}$ aller Prädikatensymbole. Als Menge *Dom* der Konstantenzeichen von $\mathscr{L}$, die gemäß Kapitel 5.1.1 unendlich groß zu wählen ist, soll hier die Vereinigung der Domänen aller Attribute aus A_R gewählt werden. Dabei wird hier davon ausgegangen, dass diese Domänen alle unendlich viele Konstantenzeichen enthalten. Insbesondere stehen damit also alle Konstantenzeichen, die in der ursprünglichen Relationeninstanz r vorkommen, auch in $\mathscr{L}$ zum Aufbau prädikatenlogischer Formeln zur Verfügung.

Aufbauend auf diesen Überlegungen können nun Ideen entwickelt werden, wie eine gegebene vollständige Relationeninstanz r über einem Relationenschema $\langle R|A_R| \quad \rangle$ als eine logik-orientierte, unvollständige Datenbankinstanz über $\mathscr{L}$ modelliert werden kann. Diesbezüglich soll erst einmal das explizit in r enthaltene, positive Wissen in einer Menge db_r^+ logik-orientiert modelliert werden. Dies kann erreicht werden, indem db_r^+ für jedes Tupel $\mu \in r$, das für jedes $i \in \{1, \ldots, |A_R|\}$ das Attribut $a_i \in A_R$ auf einen Wert v_i abbildet, eine prädikatenlogische Formel $R(v_1, \ldots, v_{|A_R|})$ enthält. Aufgrund dessen muss für eine mögliche DB-Interpretation $\mathcal{I}$, die db_r^+ erfüllen soll, stets $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ gelten. Damit kann der positive Teil des in r enthaltenen Wissens folgendermaßen formalisiert werden:

Definition 5.4 (Modellierung des positiven Wissens) *Sei eine vollständige Relationeninstanz r zu dem Relationenschema $\langle R|A_R| \quad \rangle$ mit der Attributmenge $A_R = \{a_1, \ldots, a_{|A_R|}\}$ gegeben. Das positive, explizit in r enthaltene Wissen kann in Form einer Menge*

$$db_r^+ := \left\{ R(v_1, \ldots, v_{|A_R|}) \mid \exists \mu \in r : \bigwedge_{i=1}^{|A_R|} \mu[a_i] = v_i \right\}$$

von Formeln aus der prädikatenlogischen Sprache $\mathscr{L}$ modelliert werden.

Damit kann das in r enthaltene positive Wissen prädikatenlogisch modelliert werden. Da es sich bei r aber nach Voraussetzung um eine vollständige Relationeninstanz handelt, ist in r neben diesem explizit aufgezählten Wissen noch weiteres negatives Wissen vorhanden, das sich implizit aus der für vollständige Datenbankinstanzen gültigen Closed World Assumption (siehe Kapitel 5.1.2) herleiten lässt. Diese besagt, dass all die Wertekombinationen $(v_1, \ldots, v_{|A_R|})$, die zwar aufgrund des r zugrunde liegenden Relationenschemas möglich sind, aber *nicht* durch ein Tupel aus r repräsentiert werden, in r automatisch nicht gültig sind. Dieses implizit vorhandene, negative Wissen muss in einer unvollständigen, logik-orientierten Datenbankinstanz explizit modelliert werden, um dort ebenfalls als negatives Wissen vorhanden zu sein. Ansonsten würde eine geschlossene Anfrage bezüglich dieses Wissens als undefiniert beantwortet. Für die oben genannte Wertekombination $(v_1, \ldots, v_{|A_R|})$ ist eine explizite Darstellung als negatives Wissen beispielsweise durch die Formel $\neg R(v_1, \ldots, v_{|A_R|})$ der Sprache $\mathscr{L}$ möglich.

Praktisch ist es aber nicht möglich, all diese Wertekombinationen explizit in einer Formelmenge db_r^-, die das implizit vorhandene, negative Wissen aus r enthält, wie oben beschrieben zu modellieren. Im Gegensatz zu den explizit vorhandenen Wertekombinationen zur Modellierung des positiven Wissens sind die Wertekombinationen, die das aus der Closed World Assumption resultierende negative Wissen ausdrücken, nicht auf eine endliche Anzahl beschränkt. Es handelt sich dabei stets um unendlich viele Wertekombinationen, weil die Domänen der Attribute des r zugrunde liegenden Relationenschemas nach Voraussetzung stets unendliche Größe haben, während nur endliche viele Wertekombination, die positives Wissen ausdrücken, vorhanden sein können. Dies ist der Fall, weil eine Relationeninstanz nach Definition 2.2 nur endlich viele Tupel, die positives Wissen repräsentieren, enthalten darf. Da aber in Definition 5.3 gefordert wird, dass eine logik-orientierte unvollständige Datenbankinstanz in Form einer *endlichen* Menge von Formeln aus $\mathscr{L}$ konstruiert werden muss, kann das aus der Closed World Assumption resultierende negative Wissen also nicht wie oben vorgeschlagen durch explizites Aufzählen der einzelnen Wertekombinationen behandelt werden.

Abhilfe von diesem Problem schafft das aus [14, Abschnitt 4.1] bekannte Konzept des sogenannten „Completeness-Sentence“. Durch dieses Konzept wird es ermöglicht, in der logik-orientierten Sichtweise einer Datenbankinstanz die unendlich vielen Wertekombination, die bei einer unendlich großen Domäne und endlich vielen positiven Wertekombination das implizit vorhandene negative Wissen beschreiben, indirekt durch Aufzählen aller endlich vielen positiven Wertekombination aus einer relationalen Datenbankinstanz zu modellieren.

Definition 5.5 (Modellierung des negativen Wissens) *Sei eine vollständige Relationeninstanz r zu dem Relationenschema $\langle R|A_R| \quad \rangle$ mit der Attributmenge $A_R = \{a_1, \ldots, a_{|A_R|}\}$ gegeben. Das negative, in r implizit durch die Eigenschaft der Vollständigkeit enthaltene Wissen kann explizit durch*

$$db_r^- := \left\{ (\forall X_1) \ldots (\forall X_{|A_R|}) \left[\bigvee_{\mu \in r} \left(\bigwedge_{j=1}^{|A_R|} (X_j = \mu[a_j]) \right) \vee \neg R\left(X_1, \ldots, X_{|A_R|}\right) \right] \right\}$$

in der prädikatenlogischen Sprache $\mathscr{L}$ modelliert werden. Dabei sind $X_1, \ldots, X_{|A_R|}$ Variablen aus der Menge Var der Sprache $\mathscr{L}$.

Damit können die unendlich vielen Wertekombinationen, die das aus der Closed World Assumption resultierende negative Wissen aus r ausdrücken, in db_r^- durch eine einzige Formel in Form des Completeness-Sentence formuliert werden. Dieser Completeness-Sentence enthält dabei für jedes Attribut a_i aus der Attributmenge A_R des r zugrunde liegenden Relationenschemas $\langle R|A_R| \quad \rangle$ genau eine allquantifizierte Variable X_i. Für jedes Tupel $\mu \in r$ wird für die Wertekombination $(\mu[a_1], \ldots, \mu[a_{|A_R|}])$, die aufgrund der Existenz des Tupels μ in r enthalten ist, eine Konjunktion über alle Attribute $a_i \in A_R$ gebildet. In dieser Konjunktion wird mit Hilfe des ausgezeichneten Prädikatensymbols $=$ gefordert, dass der Variablen X_i jeweils genau das Konstantenzeichen zugewiesen werden soll, auf das auch Attribut a_i durch Tupel μ in r abgebildet wird. All diese Konjunktionen über die Attributwerte der Tupel aus r werden disjunktiv verknüpft und anschließend wird die so entstandene Disjunktion noch um das weitere Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$, das ausschließlich allquantifizierte Variablen als Terme enthält, ergänzt.

Die Intention hinter dieser Formel besteht darin, das in r implizit vorhandene negative Wissen als eine Art Komplement des explizit aufgezählten positiven Wissens auszudrücken. Damit der Completeness-Sentence durch eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$ erfüllt wird, muss dieser aufgrund der ausnahmslos allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ unter *allen* möglichen Ersetzungen der Variablen $X_1, \ldots, X_{|A_R|}$ durch Konstantenzeichen aus der (unendlich großen) Domäne Dom von $\mathscr{L}$ durch $\mathcal{I}$ erfüllt werden. Für eine Wertekombination $(v_1, \ldots, v_{|A_R|})$, die in einem Tupel aus r als positives Wissen enthalten ist, ist auch stets eine entsprechende Konjunktion

$$(X_1 = v_1) \wedge (X_2 = v_2) \wedge \ldots \wedge (X_{|A_R|} = v_{|A_R|})$$

in dem zu r konstruierten Completeness-Sentence als Disjunkt enthalten. Das bedeutet, dass unter der Konstantenersetzung $(X_1/v_1), \ldots, (X_{|A_R|}/v_{|A_R|})$ der Variablen $X_1, \ldots, X_{|A_R|}$ dieses Disjunkt aufgrund der Semantik des ausgezeichneten Prädikatensymbols $=$, die für alle DB-Interpretationen fest definiert ist (vgl. Definition 5.1), erfüllt wird. Als Folge dessen brauchen die restlichen Disjunkte des

Completeness-Sentence unter dieser Konstantenersetzung nicht mehr durch die betrachtete DB-Interpretation $\mathcal{I}$ erfüllt werden, um die gesamte Formel erfüllen zu können. Damit braucht unter dieser Konstantenersetzung insbesondere auch das Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$ nicht durch $\mathcal{I}$ erfüllt werden.

Wenn für die allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ hingegen eine Konstantenersetzung $(X_1/u_1), \ldots, (X_{|A_R|}/u_{|A_R|})$ betrachtet wird, für die *keine* entsprechende Wertekombination in der Relationeninstanz r existiert, gibt es in dem zu r konstruierten Completeness-Sentence auch *kein* Disjunkt in Form einer Konjunktion

$$(X_1 = u_1) \wedge (X_2 = u_2) \wedge \ldots \wedge (X_{|A_R|} = u_{|A_R|}).$$

Auf der semantischen Ebene bedeutet dies, dass der Completeness-Sentence unter der betrachteten Konstantenersetzung durch eine DB-Interpretation $\mathcal{I}$ nur dann noch erfüllt werden kann, wenn das Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$ unter der betrachteten Konstantenersetzung erfüllt wird. Demnach muss unter $\mathcal{I}$ die Formel $\neg R(u_1, \ldots, u_{|A_R|})$ gelten, damit der Completeness-Sentence – der aufgrund der ausnahmslos allquantifizierten Variablen unter allen möglichen Konstantenersetzungen für die Variablen $X_1, \ldots, X_{|A_R|}$ zu erfüllen ist – durch $\mathcal{I}$ auch unter der Konstantenersetzung $(X_1/u_1), \ldots, (X_{|A_R|}/u_{|A_R|})$ erfüllt wird. Das bedeutet, dass das in r enthaltene negative Wissen wie gewünscht durch den Completeness-Sentence zum Ausdruck kommt, weil eine DB-Interpretation $\mathcal{I}$ die Formel $\neg R(u_1, \ldots, u_{|A_R|})$ nur erfüllen kann, wenn $(u_1, \ldots, u_{|A_R|}) \notin \mathcal{I}(R)$ gilt.

Damit kann eine Relationeninstanz r nach Wahl der passenden prädikatenlogischen Sprache $\mathscr{L}$ in Form einer logik-orientierten Datenbankinstanz db_r durch $db_r := db_r^+ \cup db_r^-$ modelliert werden, wenn db_r^+ gemäß Definition 5.4 und db_r^- gemäß Definition 5.5 konstruiert wird.

Die oben vorgestellte Modellierung einer ursprünglichen Relationeninstanz in Form einer logik-orientierten Datenbankinstanz soll im Folgenden anhand eines Beispiels vertieft werden. Dies soll anhand der in Abbildung 5.2 dargestellten vollständigen Relationeninstanz *person* geschehen, der das Schema $\langle Person | A_{\text{Person}} | \quad \rangle$ zugrunde liegt. Dieses Relationenschema enthält die Attributmenge $\{\texttt{Name}, \texttt{Geburtstag}, \texttt{Plz}\}$, und die zugehörigen Domänen, aus denen die Werte für die jeweiligen Attribute gewählt werden können, seien unendlich groß gewählt.

Demnach muss die prädikatenlogische Sprache $\mathscr{L}$, die die logik-orientierte Darstellung der Relationeninstanz *person* ermöglichen soll, hier das Prädikatensymbol $Person$ enthalten. Die Domäne Dom von $\mathscr{L}$ entspricht der Vereinigung der Mengen von Konstantenzeichen, die jeweils in den (unendlich großen) Domänen der Attribute `Name`, `Geburtstag` und `Plz` des *person* zugrunde liegenden Relationenschemas vereinbart sind. Hinsichtlich der konkreten Modellierung des positiven Wissens der

Person	**Name**	**Geburtstag**	**Plz**
	Hellmann	03.01.1981	94142
	McKinley	12.02.1952	94139
	Ripley	03.01.1981	94139

Abbildung 5.2: Relationeninstanz $person$ zu Relationenschema $Person$

Relationeninstanz $person$ müssen die drei Tupel betrachtet werden, die in $person$ explizit aufgezählt sind. Für jedes dieser Tupel $\mu \in person$ muss die Formelmenge db^{+}_{person} gemäß Definition 5.4 jeweils genau eine prädikatenlogische Formel

$$Person(\mu[\texttt{Name}], \mu[\texttt{Geburtstag}], \mu[\texttt{Plz}])$$

enthalten. Deshalb ergibt sich für db^{+}_{person} die folgende Menge von Formeln:

$$\begin{array}{llll} db^{+}_{person} = \{ & Person(\text{ Hellmann}, & 03.01.1981, & 94142\ ), \\ & Person(\text{ McKinley}, & 12.02.1952, & 94139\ ), \\ & Person(\text{ Ripley}, & 03.01.1981, & 94139\ ) \quad \} \end{array}$$

Um auch das negative Wissen, das in der vollständigen Relationeninstanz $person$ implizit enthalten ist, in der logik-orientierten Datenbankinstanz zu berücksichtigen, wird dieses Wissen wie oben beschrieben in Form einer einelementigen Formelmenge db^{-}_{person} modelliert, die ausschließlich den Completeness-Sentence

$$\begin{array}{lllllll} (\forall X_N)(\forall X_G)(\forall X_P)\ [& & & & & & \\ (X_N = \text{Hellmann} & \wedge & X_G = 03.01.1981 & \wedge & X_P = 94142) & \vee & \\ (X_N = \text{McKinley} & \wedge & X_G = 12.02.1952 & \wedge & X_P = 94139) & \vee & \\ (X_N = \text{Ripley} & \wedge & X_G = 03.01.1981 & \wedge & X_P = 94139) & \vee & \\ \neg Person(X_N, X_G, X_P) & & & & &] & \end{array}$$

enthält. Dieser Completeness-Sentence enthält für jedes der Attribute des $person$ zugrunde liegenden Relationenschemas genau eine allquantifizierte Variable. Zudem existiert in diesem für jedes der drei Tupel aus $person$ genau ein Disjunkt in Form einer Konjunktion, die durch eine (beliebige) DB-Interpretation erfüllt wird, wenn *alle* Variablen unter einer Konstantenersetzung betrachtet werden, die genau einer Wertekombination eines der Tupel aus $person$ entspricht.

Aufgrund der Semantik der allquantifizierten Variablen muss der hier konstruierte Completeness-Sentence aber unter *allen* möglichen Konstantenersetzungen dieser

Variablen durch eine DB-Interpretation $\mathcal{I}$ erfüllt werden. Deshalb müssen auch Konstantenersetzungen betrachtet werden, unter denen keines der ersten drei Disjunkte erfüllt wird. Eine solche Konstantenersetzung ist beispielsweise durch

- $X_N :=$ McKinley
- $X_G :=$ 03.01.1981
- $X_P :=$ 94142

gegeben. Unter dieser Konstantenersetzung kann der konstruierte Completeness-Sentence offensichtlich nur dann durch eine DB-Interpretation $\mathcal{I}$ erfüllt werden, wenn das letzte Disjunkt

$$\neg Person(X_N, X_G, X_P)$$

dieser Formel erfüllt ist. Das bedeutet konkret, dass durch $\mathcal{I}$ die Formel

$$\neg Person(\text{McKinley}, 03.01.1981, 94142)$$

erfüllt werden muss, damit der Completeness-Sentence auch unter der dieser betrachteten Konstantenersetzung durch $\mathcal{I}$ erfüllt werden kann.

5.2.2 Modellierung der fragmentierten Relationeninstanz

Aufgrund der Überlegungen aus Kapitel 5.2.1 existiert nun ein prädikatenlogisches Framework, in dem eine vollständige Relationeninstanz r modelliert werden kann. Daran anschließend soll jetzt erläutert werden, wie Fragment-Instanzen, die zu r im Zuge des in Kapitel 3.4 vorgestellten Fragmentierungs-Ansatzes „Fragmentierung und partielle lokale Verwaltung“ gebildet wurden, in diesem logik-orientierten Framework für r modelliert werden können. Diese Art der Darstellung von zu r gebildeten Fragment-Instanzen soll formale Beweise hinsichtlich der Inferenzsicherheit dieses Fragmentierungs-Ansatzes ermöglichen.

Bei dem hier betrachteten Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ wird eine ursprüngliche Relationeninstanz r zu einem Relationenschema $\langle R|A_R|\ \ \rangle$ immer in zwei Fragment-Instanzen f_o und f_s zerlegt.[13] Dabei ist f_s eine Fragment-Instanz zu Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle$, die auch bedenkenlos auf einem nicht vertrauenswürdigen externen Server verwaltet werden kann. Bei f_o handelt es sich um eine Fragment-Instanz zu Fragment $\langle F_o|A_{F_o}|SC_{F_o}\rangle$, die ausschließlich lokal auf einem vertrauenswürdigen Client gespeichert werden darf.

Aufbauend auf den Überlegungen aus Kapitel 5.1.4, in deren Rahmen ein nicht autorisierter Betrachter als potentieller Angreifer ausfindig gemacht wird, wird im

[13] Der Index o steht dabei für „Owner“ und der Index s steht dabei für „Server“.

Folgenden davon ausgegangen, dass ein potentieller Angreifer stets Zugriff auf die zu r entstandene Fragment-Instanz f_s hat. Ebenso sind ihm das r zugrunde liegende Relationenschema $\langle R|A_R|\ \ \rangle$ und die beiden zu diesem Schema berechneten Fragmente $\langle F_o|A_{F_o}|SC_{F_o}\rangle$ und $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ der Fragmentierung $\mathcal{F}$ bekannt. Nicht bekannt sind ihm hingegen die ursprüngliche Relationeninstanz r sowie die zu r entstandene Fragment-Instanz f_o, die ausschließlich auf dem vertrauenswürdigen Client verwaltet werden darf und deshalb nur von autorisierten Benutzern eingesehen werden kann.

Aufgrund seiner Kenntnis über das verwendete Verfahren zur Fragmentierung von r weiß ein potentieller Angreifer, dass die Fragment-Instanz f_s – abgesehen von den in f_s enthaltenen Tupel-IDs – einer Projektion der ursprünglichen Relationeninstanz r auf die Attributmenge A_{F_s} des Fragments $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ entspricht (vgl. Definition 3.7). Deshalb muss für jedes Tupel $\nu \in f_s$ auch ein Tupel $\mu \in r$ existieren, so dass für diese beiden Tupel die Beziehung $\mu\lceil A_{F_s} = \nu\lceil A_R$ gilt, wenn mit $\mu\lceil A$ die Einschränkung eines Tupels μ auf eine Attributmenge A bezeichnet wird (siehe [7, Kap. 8.1]). Aufgrund der in f_s zum Einsatz kommenden Tupel-IDs, die in SC_{F_s} als Primärschlüssel vereinbart sind und damit verhindern, dass zwei verschiedene Tupel μ_1 und μ_2 aus r aufgrund der Eigenschaft $\mu_1\lceil A_{F_s} = \mu_2\lceil A_{F_s}$ in f_s zu einem Tupel durch Duplikats-Entfernung verschmolzen werden, weiß ein potentieller Angreifer sogar, dass für jedes Tupel $\nu \in f_s$ jeweils *genau* ein Tupel $\mu \in r$ existiert, so dass für diese beiden Tupel die Beziehung $\mu\lceil A_{F_s} = \nu\lceil A_R$ gilt.

Einem potentiellen Angreifer ist also bewusst, dass ihm aufgrund seiner Kenntnis von f_s für ein Tupel $\mu \in r$ ausschließlich die Attributwerte, die durch μ den Attributen aus $A_{F_s} \cap A_R$ zugeordnet werden, bekannt sind. Die Attributwerte, die durch μ den Attributen aus $A_R \setminus A_{F_s}$ zugeordnet werden, bleiben ihm hingegen vorenthalten. Er weiß aufgrund der Vollständigkeit von r aber, dass μ einem Attribut $a_i \in A_R \setminus A_{F_s}$ stets einen Wert aus der Domäne von a_i zuordnen muss und kann deshalb von der Existenz eines solchen Wertes für a_i in μ ausgehen.

Aufgrund der Tatsache, dass zwei verschiedene Tupel aus r nicht identisch sein dürfen und bedingt durch die Tupel-IDs in f_s keine Duplikats-Entfernung vorgenommen wird (siehe oben), müssen sich zwei verschiedene Tupel aus r, die hinsichtlich der Werte der Attribute aus $A_{F_s} \cap A_R$ identisch sind, in mindestens einem der Werte der Attribute aus $A_R \setminus A_{F_s}$ unterscheiden. Da hier aber kombinatorische Effekte, die unter Ausnutzung dieses Wissens Inferenzen ermöglichen können, aufgrund der Wahl unendlich großer Domänen ausgeschlossen sind (vgl. Kapitel 5.1.1), wird hier unter der Annahme gearbeitet, dass dieses Wissen im Rahmen der Modellierung von f_s nicht weiter beachtet werden muss.

Auf Basis dieser Ideen kann in Anlehnung an die Konzepte aus Kapitel 5.2.1 das positive Wissen, das in einer Fragment-Instanz f_s bezüglich einer ursprünglichen

Relationeninstanz r explizit enthalten ist, logik-orientiert im Framework der ursprünglichen Relationeninstanz r als möglicherweise unvollständige Datenbankinstanz modelliert werden. Das bedeutet insbesondere, dass die prädikatenlogische Sprache $\mathscr{L}$, auf der die logik-orientierte Darstellung basiert, hier in derselben Ausprägung wie auch in Kapitel 5.2.1 zum Einsatz kommt.

Definition 5.6 (Modellierung des positiven Wissens) *Gegeben sei das Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ einer gemäß Definition 3.7 aufgebauten Fragmentierung $\mathcal{F}$ eines Relationenschemas $\langle R|A_R|\ \ \rangle$ und es gelte*

$$A_{F_s} \cap A_R = \{a_{i_1}, \ldots, a_{i_k}\} \subseteq \{a_1, \ldots, a_{|A_R|}\} = A_R.$$

Sei des Weiteren f_s eine Fragment-Instanz zu Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle$ bezüglich einer ursprünglichen Relationeninstanz r zu Schema $\langle R|A_R|\ \ \rangle$. Wenn

- $\mathrm{Ind}^+_{F_s} = \{i_1, \ldots, i_k\}$ *die Indizes aller Attribute aus $A_{F_s} \cap A_R$ und*
- $\mathrm{Ind}^-_{F_s} = \{1, \ldots, |A_R|\} \setminus \{i_1, \ldots, i_k\} = \{i_{k+1}, \ldots, i_{|A_R|}\}$ *die Indizes aller Attribute aus $A_R \setminus A_{F_s}$*

enthält, kann das in f_s explizit enthaltene, positive Wissen in Form einer Menge $db^+_{f_s}$ modelliert werden, die für jedes Tupel $\mu \in f_s$ genau eine Formel

$$(\exists X_{i_{k+1}}) \ldots (\exists X_{i_{|A_R|}})\ R(t_1, \ldots, t_{|A_R|})$$

aus $\mathscr{L}$ enthält. Dabei gilt für Indizes $j \in \mathrm{Ind}^+_{F_s}$ jeweils $t_j = \mu[a_j]$ und für Indizes $j \in \mathrm{Ind}^-_{F_s}$ jeweils $t_j = X_j$, wobei X_j eine Variable aus der Menge Var von $\mathscr{L}$ ist. Weitere Formeln existieren in $db^+_{f_s}$ nicht.

Damit kann der positive Teil einer Fragment-Instanz f_s also ähnlich wie der positive Teil der zugehörigen ursprünglichen Relationeninstanz r (vgl. Definition 5.4) in einem logik-orientierten Framework für r dargestellt werden, indem das in einem Tupel aus f_s enthaltene Wissen jeweils durch eine Formel aus $\mathscr{L}$ in $db^+_{f_s}$ repräsentiert wird. Dabei wird ein Wert, den ein Tupel $\mu \in f_s$ einem Attribut $a_j \in A_{F_s} \cap A_R$ zuordnet, in der zu μ gebildeten Formel durch den Term t_j in Form des Konstantenzeichens der Domäne Dom aus $\mathscr{L}$, das dem Wert $\mu[a_j]$ entspricht, dargestellt. Für Attribute aus $A_R \setminus A_{F_s}$ kennt ein potentieller Angreifer aufgrund seiner eingeschränkten Sichtweise von r, die durch die Fragment-Instanz f_s gegeben ist, hingegen keine konkreten Attributwerte. Da er jedoch aufgrund seines Wissens über das Relationenschema, über dem die vollständige Relationeninstanz r gebildet ist, weiß, dass diese Werte existieren müssen, wird der Wert eines Attributs $a_j \in A_R \setminus A_{F_s}$ in

der zu Tupel μ gebildeten Formel als Term t_j in Form einer existenzquantifizierten Variablen X_j dargestellt.

Auf Ebene der Semantik kann eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$ eine nach Definition 5.6 gebildete Formel Φ der Form

$$(\exists X_{i_{k+1}}) \ldots (\exists X_{i_{|A_R|}})\ R(t_1, \ldots, t_{|A_R|})$$

nur dann erfüllen, wenn dazu in $\mathcal{I}$ auch ein geeignetes Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ existiert. Ein solches Tupel muss für einen Term t_j von Φ in Form eines Konstantenzeichens $\tilde{v}_j$ (es gilt also $j \in \mathrm{Ind}^+_{F_s}$) auch den entsprechenden Wert $\tilde{v}_j$ als seine j-te Komponente enthalten. Es muss also

$$(v_1, \ldots, \tilde{v}_j, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$$

gelten, damit die Formel Φ durch die DB-Interpretation $\mathcal{I}$ erfüllt werden kann.

Wenn in einer DB-Interpretation $\mathcal{I}$ ein solches Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ gefunden werden kann, für das hinsichtlich *aller* Terme t_j von Φ mit Index $j \in \mathrm{Ind}^+_{F_s}$ jeweils $t_j = v_j$ gilt, wird Φ durch $\mathcal{I}$ erfüllt: Nach Kapitel 5.1.1 kann Φ durch $\mathcal{I}$ erfüllt werden, wenn für die existenzquantifizierten Variablen aus Φ eine beliebige Konstantenersetzung gefunden werden kann, unter der Φ durch $\mathcal{I}$ erfüllt wird. Wenn also für jedes $j \in \mathrm{Ind}^-_{F_s}$ die existenzquantifizierte Variable X_j aus Φ durch v_j aus dem betrachteten Tupel $(v_1, \ldots, v_{|A_R|})$ ersetzt wird, stimmt für jedes $h \in \{1, \ldots, |A_R|\}$ jeweils der Wert der h-ten Komponente des Tupels $(v_1, \ldots, v_{|A_R|})$ mit dem Konstantenzeichen des Terms t_h überein. Als Folge dessen gilt $\mathcal{I} \models_M \Phi$.

Neben diesem positiven Wissen muss aber auch hier bei der logik-orientierten Modellierung einer Fragment-Instanz f_s im Kontext der ursprünglichen Relationeninstanz r das negative Wissen betrachtet werden, das sich aufgrund der Eigenschaft der Vollständigkeit implizit aus der Closed World Assumption ergibt. Eine aufgrund der zugrunde liegenden Domänen mögliche Wertekombination $(v_{i_1}, \ldots, v_{i_k})$, die bezüglich der Werte der Attribute aus $A_{F_s} \cap A_R$ durch *kein* Tupel der Fragment-Instanz f_s repräsentiert wird, kann auch in *keinem* Tupel der ursprünglichen Relationeninstanz r, zu der f_s gebildet wurde, durch die Werte der Attribute aus $A_{F_s} \cap A_R$ repräsentiert werden. Andernfalls müsste gemäß der Konstruktion der Fragment-Instanz f_s nach Definition 3.7 auch ein entsprechendes Tupel, das eben diese Wertekombination $(v_{i_1}, \ldots, v_{i_k})$ durch die Werte der Attribute aus $A_{F_s} \cap A_R$ repräsentiert, in f_s enthalten sein.

Das bedeutet, dass das implizit vorhandene negative Wissen, das aus einer solchen Wertekombination $(v_{i_1}, \ldots, v_{i_k})$ resultiert, in der prädikatenlogischen Sprache $\mathscr{L}$ explizit durch eine Formel

$$(\forall X_{i_{k+1}}) \ldots (\forall X_{i_{|A_R|}})\ \neg R(t_1, \ldots, t_{|A_R|})$$

modelliert werden kann. In dieser Formel ist für jedes $j \in \mathrm{Ind}^+_{F_s}$ der Term t_j in Form des Konstantenzeichens aus $\mathscr{L}$, das dem Wert v_j der betrachteten Wertekombination $(v_{i_1}, \ldots, v_{i_k})$ entspricht, vorhanden. Ein Term t_j mit $j \in \mathrm{Ind}^-_{F_s}$ ist hingegen in Form der allquantifizierten Variablen X_j gegeben. Dabei sind die Indexmengen $\mathrm{Ind}^+_{F_s}$ und $\mathrm{Ind}^-_{F_s}$ wie in Definition 5.6 aufgebaut. Die Wahl allquantifizierter Variablen für die Terme $t_{i_{k+1}}, \ldots, t_{i_{|A_R|}}$ der oben dargestellten Formel ist dabei folgendermaßen begründet: Da die Wertekombination $(v_{i_1}, \ldots, v_{i_k})$ hinsichtlich der Werte der Attribute aus $A_{F_s} \cap A_R$ durch *kein* Tupel aus r repräsentiert wird, kann es auch kein Tupel in r geben, das diese Wertekombination für die Werte der Attribute aus $A_{F_s} \cap A_R$ zusammen mit einer beliebigen anderen Wertekombination $(v_{i_{k+1}} \ldots v_{i_{|A_R|}})$ für die Werte der Attribute aus $A_R \setminus A_{F_s}$ enthält.

Analog zu der in Kapitel 5.2.1 beschriebenen Modellierung des negativen Wissens der ursprünglichen Relationeninstanz r ergibt sich aber auch hier das Problem, dass es unendlich viele mögliche Wertekombinationen gibt, die bezüglich der Werte der Attribute aus $A_{F_s} \cap A_R$ durch *kein* Tupel der Fragment-Instanz f_s repräsentiert werden. Deshalb ist es auch hier nicht möglich, das komplette negative Wissen, das sich durch Anwendung der Closed World Assumption herleiten lässt, durch explizites Aufzählen der einzelnen Wertekombinationen, die in f_s durch kein Tupel aus $A_{F_s} \cap A_R$ repräsentiert werden, zu modellieren. Aufgrund dessen soll das implizit vorhandene negative Wissen auch hier in Anlehnung an [14, Abschnitt 4.1] durch einen Completeness-Sentence modelliert werden.

Definition 5.7 (Modellierung des negativen Wissens) *Gegeben sei das Fragment $\langle F_s | A_{F_s} | SC_{F_s} \rangle$ einer gemäß Definition 3.7 aufgebauten Fragmentierung $\mathcal{F}$ eines Relationenschemas $\langle R | A_R | \quad \rangle$ und es gelte*

$$A_{F_s} \cap A_R = \{a_{i_1}, \ldots, a_{i_k}\} \subseteq \{a_1, \ldots, a_{|A_R|}\} = A_R.$$

Sei des Weiteren f_s eine Fragment-Instanz zu Fragment $\langle F_s | A_{F_s} | SC_{F_s} \rangle$ bezüglich einer ursprünglichen Relationeninstanz r zu Schema $\langle R | A_R | \quad \rangle$. Wenn die Indexmenge $\mathrm{Ind}^+_{F_s} = \{i_1, \ldots, i_k\}$ die Indizes aller Attribute aus $A_{F_s} \cap A_R$ enthält, kann das in f_s implizit enthaltene, negative Wissen durch eine Menge $db^-_{f_s}$, die ausschließlich den Completeness-Sentence

$$(\forall X_1) \ldots \left(\forall X_{|A_R|}\right) \left[\bigvee_{\mu \in f_s} \left(\bigwedge_{j \in \mathrm{Ind}^+_{F_s}} (X_j = \mu[a_j]) \right) \vee \neg R\left(X_1, \ldots, X_{|A_R|}\right) \right]$$

enthält, in der prädikatenlogischen Sprache $\mathscr{L}$ modelliert werden. Dabei sind $X_1, \ldots, X_{|A_R|}$ Variablen aus der Menge Var der Sprache $\mathscr{L}$.

Die Konstruktion dieses Completeness-Sentence[14] verläuft ähnlich zu der Konstruktion des Completeness-Sentence aus Definition 5.5 zur Modellierung des negativen Wissens der ursprünglichen Relationeninstanz r. Wie auch dort enthält dieser Completeness-Sentence für jedes Attribut a_j aus der Attributmenge A_R des Relationenschemas $\langle R|A_R| \quad \rangle$, das der ursprünglichen Relationeninstanz r zugrunde liegt, genau eine allquantifizierte Variable X_j. Für jedes Tupel μ aus der zu r gebildeten Fragment-Instanz f_s wird über alle Komponenten der Wertekombination $(\mu[a_{i_1}], \ldots, \mu[a_{i_k}])$, die genau die Attributwerte enthält, die μ den Attributen aus $A_{F_s} \cap A_R$ zuordnet, eine Konjunktion gebildet.

In dieser Konjunktion wird für alle Indizes $j \in \mathrm{Ind}^+_{F_s}$ gefordert, dass der Variablen X_j jeweils genau das Konstantenzeichen zugeordnet werden soll, das dem Wert entspricht, auf den auch Attribut $a_j \in A_{F_s} \cap A_R$ durch das betrachtete Tupel $\mu \in f_s$ abgebildet wird. Die Konstruktion dieser Konjunktion ist dabei stets möglich, weil es sich bei der Menge der Attribute aus $A_{F_s} \cap A_R$, über deren Indizes die betrachtete Konjunktion gebildet wird, um eine Teilmenge der Attributmenge A_R handelt, zu der für jedes Attribut dieser Menge genau eine allquantifizierte Variable in dem Completeness-Sentence existiert. Wie auch in Definition 5.5 werden die so gebildeten Konjunktionen disjunktiv verknüpft und die dadurch entstandene Disjunktion noch um das weitere Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$, das ausschließlich allquantifizierte Variablen als Terme enthält, ergänzt.

Damit der Completeness-Sentence durch eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$ erfüllt wird, muss dieser (wie auch in Kapitel 5.2.1) aufgrund der ausnahmslos allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ unter *allen* möglichen Ersetzungen der Variablen $X_1, \ldots, X_{|A_R|}$ durch Konstantenzeichen aus der (unendlich großen) Domäne *Dom* von $\mathscr{L}$ durch $\mathcal{I}$ erfüllt werden. Für eine Wertekombination $(v_{i_1}, \ldots, v_{i_k})$, die in einem Tupel $\mu \in f_s$ durch die Werte der Attribute aus $A_{F_s} \cap A_R$ repräsentiert wird, existiert stets ein Disjunkt in Form einer Konjunktion

$$(X_{i_1} = v_{i_1}) \wedge (X_{i_2} = v_{i_2}) \wedge \ldots \wedge (X_{i_k} = v_{i_k})$$

in dem zu f_s konstruierten Completeness-Sentence. Im Falle einer Konstantenersetzung der Variablen $X_1, \ldots, X_{|A_R|}$, unter der für alle $j \in \mathrm{Ind}^+_{F_s}$ die Variable X_j jeweils durch das Konstantenzeichen v_j ersetzt wird, wird dieses Disjunkt des Completeness-Sentence aufgrund der Semantik des ausgezeichneten Prädikatensymbols $=$ unter einer (beliebigen) DB-Interpretation $\mathcal{I}$ erfüllt (vgl. Definition 5.1). Zur Erfüllung des gesamten Completeness-Sentence durch $\mathcal{I}$ brauchen damit unter dieser Konstantenersetzung die restlichen Disjunkte des Completeness-Sentence nicht

[14] Im Gegensatz zu dem Completeness-Sentence aus Definition 5.5 wird durch den hier definierten Completeness-Sentence kein vollständiges negatives Wissen über r formuliert. Die Eigenschaft der Vollständigkeit bezieht sich ausschließlich auf Wissen aus der Fragment-Instanz f_s.

F_o	**tid**	**Name**
	1	Hellmann
	2	McKinley
	3	Ripley

F_s	**tid**	**Geburtstag**	**Plz**
	1	03.01.1981	94142
	2	12.02.1952	94139
	3	03.01.1981	94139

Abbildung 5.3: Fragment-Instanzen f_o und f_s zu Relationeninstanz *person*

mehr durch $\mathcal{I}$ erfüllt zu werden. Insbesondere braucht unter dieser Konstantenersetzung also auch das Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$ nicht durch $\mathcal{I}$ erfüllt werden.

Für die allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ müssen aber auch hier Konstantenersetzungen $(X_1/u_1), \ldots, (X_{|A_R|}/u_{|A_R|})$ betrachtet werden, die die Variablen $X_{i_1}, \ldots, X_{i_k}$ derart durch Konstantenzeichen $u_{i_1}, \ldots, u_{i_k}$ aus $\mathscr{L}$ ersetzen, dass *kein* Tupel in f_s existiert, das eine entsprechende Wertekombination für die Attribute aus $A_{F_s} \cap A_R$ enthält. In diesem Fall gibt es in dem zu f_s konstruierten Completeness-Sentence auch *kein* Disjunkt in Form einer Konjunktion

$$(X_{i_1} = u_{i_1}) \wedge (X_{i_2} = u_{i_2}) \wedge \ldots \wedge (X_{i_k} = u_{i_k}),$$

die unter dieser Konstantenersetzung durch eine (beliebige) DB-Interpretation $\mathcal{I}$ erfüllt wird. Damit kann der Completeness-Sentence unter einer solchen Konstantenersetzung durch eine DB-Interpretation $\mathcal{I}$ nur dann erfüllt werden, wenn durch $\mathcal{I}$ das Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$ unter dieser Konstantenersetzung erfüllt wird. Dazu muss $(u_1, \ldots, u_{|A_R|}) \notin \mathcal{I}(R)$ gelten, so dass auch hier der gebildete Completeness-Sentence das negative Wissen, das ein potentieller Angreifer aus der Fragment-Instanz f_s herleiten kann, beschreibt.

Damit kann eine Fragment-Instanz f_s, die zu einer ursprünglichen Relationeninstanz r gebildet wurde, in einer logik-orientierten Datenbankinstanz db_{f_s} im Kontext von r durch $db_{f_s} := db^+_{f_s} \cup db^-_{f_s}$ modelliert werden. Dabei wird $db^+_{f_s}$ gemäß Definition 5.6 und $db^-_{f_s}$ gemäß Definition 5.7 konstruiert.

Auch hier soll die oben vorgestellte logik-orientierte Modellierung einer Fragment-Instanz f_s anhand eines Beispiels verdeutlicht werden. Dazu wird wieder die aus Abbildung 5.2 bekannte Relationeninstanz *person*, die zu dem Relationenschema $\langle Person | A_{\text{Person}} | \ \rangle$ gebildet ist, aufgegriffen. Unter den Vertraulichkeitsanforderungen in Form der Vertraulichkeits-Constraints

- $c_0 = \{\texttt{Name}, \texttt{Geburtstag}\}$
- $c_1 = \{\texttt{Name}, \texttt{Plz}\}$

können sich bei dem Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ die beiden Fragment-Instanzen f_o und f_s ergeben, die in Abbildung 5.3 zu sehen sind und als Grundlage für die nachfolgenden Beispiele dienen sollen. Dabei wird davon ausgegangen, dass f_s so gespeichert wird, dass ein potentieller Angreifer Zugriff diese Fragment-Instanz hat, während f_o so verwaltet wird, dass diese Fragment-Instanz sicher vor unbefugtem Zugriff ist.

Durch seine Kenntnis von f_s kennt ein potentieller Angreifer alle Wertekombinationen für die Attribute `Geburtstag` und `Plz`, die auch in einem Tupel der ursprünglichen Relationeninstanz *person* vorkommen. Aufgrund seiner Kenntnis des *person* zugrunde liegenden Relationenschemas $\langle Person|A_{\text{Person}}| \quad \rangle$ weiß er des Weiteren, dass zu jeder dieser Wertekombinationen ein dritter Wert für das Attribut `Name` gehört, der ihm aber jeweils vorenthalten bleibt. Damit kann das in der Fragment-Instanz f_s explizit enthaltene, positive Wissen im Kontext der ursprünglichen Relationeninstanz *person* gemäß Definition 5.6 modelliert werden, indem für jedes Tupel $\mu \in f_s$ jeweils eine prädikatenlogische Formel

$$(\exists X_N)\ Person(X_N, \mu[\texttt{Geburtstag}], \mu[\texttt{Plz}])$$

gebildet wird, in der das fehlende Wissen über den Wert des Attributs `Name` in μ durch die existenzquantifizierte Variable X_N zum Ausdruck kommt. Damit ergibt sich für $db^+_{f_s}$ die folgende Menge von Formeln, in der sich die Sichtweise eines potentiellen Angreifers auf die ursprüngliche Relationeninstanz *person* durch seine Kenntnis von f_s widerspiegelt:

$$\begin{array}{lllll} db^+_{f_s} = \{ & (\exists X_N)\ Person(\ X_N, & 03.01.1981, & 94142\ ), & \\ & (\exists X_N)\ Person(\ X_N, & 12.02.1952, & 94139\ ), & \\ & (\exists X_N)\ Person(\ X_N, & 03.01.1981, & 94139\ ) & \} \end{array}$$

Um auch das negative Wissen, das ein potentieller Angreifer aufgrund seiner Kenntnis der Fragment-Instanz f_s bezogen auf die ursprüngliche Relationeninstanz *person* herleiten kann, in der logik-orientierten Datenbankinstanz zu berücksichtigen, wird gemäß Definition 5.7 die einelementige Formelmenge $db^-_{f_s}$, die ausschließlich den Completeness-Sentence

$$\begin{array}{llll} (\forall X_N)(\forall X_G)(\forall X_P)\ [& & & \\ (X_G = 03.01.1981 & \wedge & X_P = 94142) & \vee \\ (X_G = 12.02.1952 & \wedge & X_P = 94139) & \vee \\ (X_G = 03.01.1981 & \wedge & X_P = 94139) & \vee \\ \neg Person(X_N, X_G, X_P) & & &] \end{array}$$

enthält, konstruiert. Dieser enthält für jedes Tupel aus f_s ein Disjunkt in Form einer Konjunktion, durch das der Completeness-Sentence unter einer beliebigen DB-Interpretation $\mathcal{I}$ für bestimmte Konstantenersetzungen der allquantifizierten Variablen X_G und X_P erfüllt werden kann. Dazu muss für X_G und X_P, die den Attributen `Geburtstag` und `Plz` des f_s zugrunde liegenden Relationenschemas entsprechen, eine Konstantenersetzung betrachtet werden, für die in einem Tupel von f_s auch eine entsprechende Wertekombination für die Attribute `Geburtstag` und `Plz` repräsentiert wird.

Auch hier müssen aber aufgrund der Semantik der allquantifizierten Variablen X_N, X_G und X_P auch Konstantenersetzungen betrachtet werden, unter denen keines der ersten drei Disjunkte des zu f_s konstruierten Completeness-Sentence erfüllt wird. Eine solche Konstantenersetzung ist beispielsweise durch

- $X_N :=$ Hellmann
- $X_G :=$ 12.02.1952
- $X_P :=$ 94142

gegeben. Da unter dieser Konstantenersetzung keines der ersten drei Disjunkte des zu f_s konstruierten Completeness-Sentence erfüllt werden kann, kann dieser Completeness-Sentence unter der gegebenen Konstantenersetzung durch eine DB-Interpretation $\mathcal{I}$ nur dann erfüllt werden, wenn das Disjunkt

$$\neg Person(X_N, X_G, X_P)$$

des Completeness-Sentence unter der betrachteten Konstantenersetzung durch die DB-Interpretation $\mathcal{I}$ erfüllt wird. Damit dies der Fall ist, muss

$$(\text{Hellmann}, 12.02.1952, 94142) \notin \mathcal{I}(Person)$$

für die DB-Interpretation $\mathcal{I}$ gelten.

Bei dem zu der Fragment-Instanz f_s konstruierten Completeness-Sentence ist entscheidend, dass sich die Konjunktionen der ersten drei Disjunkte jeweils ausschließlich auf die Variablen X_G und X_P beziehen. Diese beiden Variablen entsprechen mit den Attributen `Geburtstag` und `Plz` genau der Attributmenge $A_{F_s} \cap A_{\text{Person}}$. Aus diesem Grund kann der zu f_s konstruierte Completeness-Sentence im Falle der aus Kapitel 5.2.1 bekannten Konstantenersetzung

- $X_N :=$ McKinley
- $X_G :=$ 03.01.1981
- $X_P :=$ 94142

hier auch durch das Disjunkt

$$X_G = 03.01.1981 \ \wedge \ X_P = 94142$$

des Completeness-Sentence erfüllt werden, ohne dass dazu wie in dem Beispiel aus Kapitel 5.2.1 unter dieser Konstantenersetzung das Disjunkt

$$\neg Person(X_N, X_G, X_P)$$

des Completeness-Sentence durch $\mathcal{I}$ erfüllt werden braucht. Da sich die Konjunktion aus dem betrachteten Disjunkt

$$X_G = 03.01.1981 \ \wedge \ X_P = 94142$$

aber ausschließlich auf die Variablen X_G und X_P bezieht, die den Attributen `Geburtstag` und `Plz` der Attributmenge $A_{F_s} \cap A_{\text{Person}}$ entsprechen, kann der konstruierte Completeness-Sentence auch dann durch dieses Disjunkt erfüllt werden, wenn in der betrachteten Konstantenersetzung die Variable X_P durch eine beliebige andere Konstante (z.B. `Maier`) ersetzt wird.

5.2.3 Formalisierung der Vertraulichkeitspolitik

In Kapitel 5.1.3 stellt es sich als zweckdienlich heraus, die Vertraulichkeitsanforderungen, die im Rahmen der Fragmentierungs-Ansätze durch Vertraulichkeits-Constraints definiert werden können, im Framework der kontrollierten Anfrageauswertung durch potentielle Geheimnisse auszudrücken. Die Wahl dieses Typs Vertraulichkeitspolitik liegt dabei in dessen Semantik, die sich mit der (angenommenen) Semantik von Vertraulichkeits-Constraints überdeckt, begründet: Sowohl durch Vertraulichkeits-Constraints als auch durch potentielle Geheimnisse soll verhindert werden, dass ein dazu nicht autorisierter Benutzer in Erfahrung bringen kann, dass ein Element, welches gemäß der jeweiligen Vertraulichkeitspolitik zu schützen ist, in der jeweils betrachteten Datenbankinstanz gültig ist.

Ein Vertraulichkeits-Constraint $c_i = \{a_{i_1}, \ldots, a_{i_k}\}$ besteht gemäß Definition 3.2 aus einer Teilmenge der Attribute eines Relationenschemas $\langle R|A_R| \ \rangle$. Für eine Relationeninstanz r, die zu dem Relationenschema $\langle R|A_R| \ \rangle$ gebildet ist, drückt das Vertraulichkeits-Constraint c_i aus, dass die Kombination der Werte, die durch ein beliebiges Tupel $\mu \in r$ den Attributen $a_{i_1}, \ldots, a_{i_k} \in c_i \subseteq A_R$ zugeordnet werden, für einen potentiellen Angreifer nicht vollständig sichtbar sein darf. Da potentielle Geheimnisse einzelne zu schützende Informations-Aspekte definieren, die jeweils durch eine Formel aus der prädikatenlogischen Sprache $\mathscr{L}$ beschrieben werden, muss dementsprechend auch jede Kombination von Werten bestimmter Attribute, die in

einem Tupel aus r enthalten ist und aufgrund eines Vertraulichkeits-Constraints zu schützen ist, jeweils durch ein potentielles Geheimnis als ein zu schützender Informations-Aspekt deklariert werden.

Die Werte, die in der Relationeninstanz r durch μ den Attributen $a_{i_{k+1}}, \ldots, a_{i_{|A_R|}}$ aus der Menge $A_R \setminus c_i$ zugeordnet werden, sind dabei durch das Vertraulichkeits-Constraint c_i weder in dieser Kombination, noch einzeln zu schützen. Allerdings können diese Werte in einer Formel aus $\mathscr{L}$, die ein potentielles Geheimnis darstellt, trotzdem nicht vollständig vernachlässigt werden: Das zu Schema $\langle R|A_R| \quad \rangle$ in der Sprache $\mathscr{L}$ existierende Prädikatensymbol R hat die Stelligkeit $|A_R|$ und damit steht für jedes Attribut a_j aus A_R jeweils genau ein Term t_j in einer entsprechenden Formel aus $\mathscr{L}$ zur Verfügung, der in einer Formel aus $\mathscr{L}$ auch definiert sein muss.

Basierend auf diesen Überlegungen kann ein erster Versuch unternommen werden, die Wertekombinationen einer Relationeninstanz r zu einem Schema $\langle R|A_R| \quad \rangle$, die gemäß eines gegebenen Vertraulichkeits-Constraints $c_i = \{a_{i_1}, \ldots, a_{i_k}\}$ zu schützen sind, jeweils in Form eines potentiellen Geheimnisses auszudrücken. Dazu sei für eine solche Wertekombination $(\mu[a_{i_1}], \ldots, \mu[a_{i_k}])$ die Formel

$$(\exists X_{i_{k+1}}) \ldots (\exists X_{i_{|A_R|}})\ R(t_1, \ldots, t_{|A_R|})$$

als potentielles Geheimnis gegeben, in der für jedes Attribut $a_j \in c_i$ der zugehörige Term t_j durch das Konstantenzeichen aus $\mathscr{L}$ repräsentiert wird, das dem Attributwert $\mu[a_j]$ entspricht. Für jedes Attribut

$$a_h \in A_R \setminus c_i = \{a_{i_{k+1}}, \ldots, a_{i_{|A_R|}}\}$$

besteht der zugehörige Term t_h hingegen aus der existenzquantifizierten Variablen X_h, die aus der Menge Var der Variablen der Sprache $\mathscr{L}$ stammt.

Hinsichtlich der Semantik eines derart konstruierten potentiellen Geheimnisses Ψ gilt damit – analog zu den Überlegungen bezüglich der Semantik einer gemäß Definition 5.6 gebildeten Formel – folgendes: Wenn in einer DB-Interpretation $\mathcal{I}$ ein Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ existiert, für das für jeden Term t_j der Formel Ψ, der den Wert eines Attributs $a_j \in c_i$ in Form eines Konstantenzeichens repräsentiert, jeweils $t_j = v_j$ gilt, kann in Ψ für jedes Attribut $a_h \in A_R \setminus c_i$ die entsprechende existenzquantifizierte Variable X_h durch v_h aus dem betrachteten Tupel $(v_1, \ldots, v_{|A_R|})$ ersetzt werden. Unter dieser Konstantenersetzung wird die betrachtete Formel Ψ durch $\mathcal{I}$ erfüllt. Deshalb kann Ψ durch eine DB-Interpretation $\mathcal{I}$ stets erfüllt werden, sofern es ein Tupel $(v_1, \ldots, v_{|A_R|}) \in \mathcal{I}(R)$ gibt, in dem für jedes Attribut $a_j \in c_i$ die j-te Komponente des Tupels $(v_1, \ldots, v_{|A_R|})$ mit dem Konstantenzeichen, das dem Term t_j von Ψ entspricht, übereinstimmt. Das bedeutet, dass die Entscheidung, ob

ein derart konstruiertes potentielles Geheimnis durch eine DB-Interpretation $\mathcal{I}$ erfüllt wird, wie gewünscht ausschließlich von den Termen, die Werten der Attribute aus c_i entsprechen, abhängig ist.

Allerdings darf die Menge der potentiellen Geheimnisse nicht so aufgebaut werden, dass diese ausschließlich potentielle Geheimnisse für die zu schützenden Wertekombinationen enthält, die auch in der betrachteten Relationeninstanz r enthalten sind. Ein potentieller Angreifer kennt nach den Überlegungen aus Kapitel 5.1.4 die verwendete Vertraulichkeitspolitik und ihm ist aufgrund seines Wissens über das verwendete Framework auch bekannt, dass in diesem Fall die Menge der potentiellen Geheimnisse aufgrund ihrer Konstruktion ausschließlich Wertekombinationen enthält, die in der ursprünglichen Relationeninstanz r gültig sind. Damit könnte ein potentieller Angreifer die Wertekombinationen, die ihm eigentlich vorenthalten bleiben sollten, unmittelbar aus der Menge der potentiellen Geheimnisse auslesen.

Die Möglichkeit des unmittelbaren Auslesens geschützter gültiger Wertekombinationen muss also unterbunden werden. Dies kann erreicht werden, indem bei einer gegebenen Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für jedes Vertraulichkeits-Constraint $c_i = \{a_{i_1}, \ldots, a_{i_k}\}$ aus $\mathcal{C}$ jede Kombinationen der Werte der Attribute $a_{i_1}, \ldots, a_{i_k}$, die aufgrund der Domänen dieser Attribute möglich ist, durch ein potentielles Geheimnis geschützt wird. Dadurch wird erreicht, dass genau die Wertekombinationen, die in einer betrachteten Relationeninstanz r aufgrund von Vertraulichkeits-Constraints aus $\mathcal{C}$ zu schützen sind, auch geschützt werden: Gemäß der Semantik potentieller Geheimnisse werden so genau die Wertekombinationen geschützt, die auch in r gültig sind.

Für die restlichen Wertekombinationen, die in r nicht gültig sind, gelten hingegen trotz der für sie definierten potentiellen Geheimnisse keine Schutzanforderungen, weil aus einem potentiellen Geheimnisses Ψ heraus nur Vertraulichkeitsanforderungen resultieren, wenn Ψ in der betrachteten Datenbankinstanz auch gültig ist. Ein potentieller Angreifer bleibt aber stets im Unklaren darüber, welche der (für ihn auslesbaren) potentiellen Geheimnisse eine Wertekombination repräsentieren, die in r gültig ist, und welche nicht (vgl. [17, Abschnitt 2.2]). Deshalb erlaubt ihm seine Kenntnis der definierten potentiellen Geheimnisse keine Rückschlüsse auf die Gültigkeit von Wertekombinationen in der Relationeninstanz r.

Das Schützen jeder möglichen Wertekombination für ein Vertraulichkeits-Constraint $c_i = \{a_{i_1}, \ldots, a_{i_k}\}$ aus $\mathcal{C}$ führt aufgrund der (nach Annahme) unendlich großen Domänen dazu, dass für jedes Vertraulichkeits-Constraint $c_i \in \mathcal{C}$ jeweils unendlich viele potentielle Geheimnisse in der prädikatenlogischen Sprache $\mathscr{L}$ definiert werden müssen. Um die Menge $pot\_sec(\mathcal{C})$ – die alle potentiellen Geheimnisse zur Modellierung der Vertraulichkeitsanforderungen aus $\mathcal{C}$ im Framework der kontrollierten Anfrageauswertung enthält – dennoch endlich groß formulieren zu können,

bietet es sich an, die in Kapitel 5.1.1 definierte prädikatenlogische Sprache $\mathscr{L}$ in Anlehnung an [16, Abschnitt 3.2.3] für die Definition der Vertraulichkeitspolitik um freie Variablen zu erweitern.

Als Folge dessen entsteht eine Sprache $\mathscr{L}^f$, für die $\mathscr{L} \subset \mathscr{L}^f$ gilt. Alle Formeln, die gemäß der Sprache $\mathscr{L}$ gebildet werden können, sind demnach auch in $\mathscr{L}^f$ enthalten. Zusätzlich sind in $\mathscr{L}^f$ aber auch Formeln enthalten, in denen Variablen aus Var frei – also weder allquantifiziert, noch existenzquantifiziert – vorkommen können. Eine Formel Ψ der Sprache $\mathscr{L}^f$, die mindestens eine freie Variable enthält, kann auch alternativ durch $\Psi(\boldsymbol{X})$ notiert werden. In dieser Notation werden durch den Vektor $\boldsymbol{X} = (X_1, \ldots, X_\ell)$ explizit die Variablen $X_1, \ldots, X_\ell$ aus der Menge Var von $\mathscr{L}^f$ angegeben, die in der Formel Ψ frei vorkommen.

Auf Basis der erweiterten prädikatenlogischen Sprache $\mathscr{L}^f$ kann nachfolgend die Modellierung einer Menge von Vertraulichkeits-Constraints des Ansatzes „Fragmentierung und partielle lokale Verwaltung“ im Framework der kontrollierten Anfrageauswertung definiert werden.

Definition 5.8 (Konstruktion der Vertraulichkeitspolitik) *Sei mit $\mathcal{C}$ eine Menge von Vertraulichkeits-Constraints gegeben, die zu einem Relationenschema $\langle R|A_R|\ \rangle$ gemäß Definition 3.2 gebildet sind. Wenn für ein Vertraulichkeits-Constraint $c_i \in \mathcal{C}$, für das $c_i = \{a_{i_1}, \ldots, a_{i_k}\} \subseteq \{a_1, \ldots, a_{|A_R|}\} = A_R$ gilt,*

- $\mathrm{Ind}^+_{c_i} = \{i_1, \ldots, i_k\}$ *die Indizes aller Attribute aus c_i und*
- $\mathrm{Ind}^-_{c_i} = \{1, \ldots, |A_R|\} \setminus \{i_1, \ldots, i_k\} = \{i_{k+1}, \ldots, i_{|A_R|}\}$ *die Indizes aller Attribute aus $A_R \setminus c_i$*

enthält, kann c_i im Framework der kontrollierten Anfrageauswertung in der erweiterten prädikatenlogischen Sprache $\mathscr{L}^f$ durch das potentielle Geheimnis

$$\Psi_i(\boldsymbol{X_i}) \ := \ (\exists X_{i_{k+1}}) \ldots (\exists X_{i_{|A_R|}})\ R(X_1, \ldots, X_{|A_R|})$$

modelliert werden. Der Vektor der Variablen aus $\mathscr{L}^f$, die in dieser Formel frei vorkommen, ist dabei durch $\boldsymbol{X_i} = (X_{i_1}, \ldots, X_{i_k})$ gegeben.
Mit $pot\_sec(\mathcal{C})$ wird die Menge von potentiellen Geheimnissen bezeichnet, die für jedes Vertraulichkeits-Constraint c_i aus $\mathcal{C}$ jeweils genau die Formel $\Psi_i(\boldsymbol{X_i})$ aus $\mathscr{L}^f$ enthält. Weitere Formeln existieren in $pot\_sec(\mathcal{C})$ nicht.

Die potentiellen Geheimnisse der Menge $pot\_sec(\mathcal{C})$, durch die die Vertraulichkeits-Constraints aus $\mathcal{C}$ im Framework der kontrollierten Anfrageauswertung beschrieben

werden können, werden dabei nach einem ähnlichen Prinzip wie in dem oben vorgestellten ersten Versuch einer prädikatenlogischen Modellierung der Vertraulichkeits-Constraints konstruiert. Auch hier wird bei der Modellierung eines Vertraulichkeits-Constraints $c_i = \{a_{i_1}, \ldots, a_{i_k}\}$ für ein Attribut $a_j \in A_R \setminus c_i$, dessen zugeordnete Werte nicht in einer durch c_i zu schützenden Wertekombination enthalten sind, in einer entsprechenden Formel $\Psi_i(\boldsymbol{X_i})$ aus $\mathscr{L}^f$ eine existenzquantifizierte Variable für den Term t_j eingesetzt. Allerdings werden hier für c_i nicht alle Wertekombinationen, die für die Attribute aus c_i aufgrund der zugrunde liegenden Domänen möglich sind, explizit aufgezählt, indem für jede dieser Wertekombinationen jeweils ein potentielles Geheimnis mit einer entsprechenden Konstantenkombination erzeugt wird.

Statt dessen wird für c_i nur ein potentielles Geheimnis $\Psi_i(\boldsymbol{X_i})$ konstruiert, in dem ein Attribut $a_j \in c_i$ durch die freie Variable X_j repräsentiert wird. Dabei kann in $\Psi_i(\boldsymbol{X_i})$ für jede freie Variable X_j (mit $j \in \mathrm{Ind}_{c_i}^+$) ein beliebiges Konstantenzeichen aus der Menge *Dom* von $\mathscr{L}^f$ eingesetzt werden. Aufgrund dessen kann für jede mögliche Wertekombination der in c_i enthaltenen Attribute $a_{i_1}, \ldots, a_{i_k}$ auch für die freien Variablen $X_{i_1}, \ldots, X_{i_k}$ der Formel $\Psi_i(\boldsymbol{X_i})$ eine entsprechende Konstantenkombination eingesetzt werden.

Im Rahmen der Konstruktion der potentiellen Geheimnisse nach Definition 5.8 ergibt sich hier aber das Problem, dass die Konzepte der DB-Interpretation (siehe Definition 5.1) und der DB-Implikation (siehe Definition 5.2) lediglich für Formeln der Sprache $\mathscr{L}$ anwendbar sind. In dieser Sprache sind im Gegensatz zu der Sprache $\mathscr{L}^f$, aus der die potentiellen Geheimnisse stammen, ausschließlich geschlossene Formeln ohne freie Variablen enthalten. Aus diesem Grund gibt es das Konzept der sogenannten Expansion von Formeln mit freien Variablen, bei dem eine Formel aus $\mathscr{L}^f$ mit Hilfe von Konstantenersetzungen in eine Menge von Formeln aus $\mathscr{L}$ transformiert werden kann [16, Abschnitt 3.2.3].

Definition 5.9 (Expansion von Formeln) *Sei eine Formel $\Psi(\boldsymbol{X})$ aus der Sprache $\mathscr{L}^f$ gegeben, die die freien Variablen $\boldsymbol{X} = (X_1, \ldots, X_\ell)$ enthält. Die Formel $\Psi(\boldsymbol{X}) \in \mathscr{L}^f$ kann durch Expansion in die Formelmenge $\mathrm{ex}(\Psi(\boldsymbol{X})) \subset \mathscr{L}$ transformiert werden, die für jede Konstantenersetzung der Variablen aus $\boldsymbol{X}$ durch eine Konstantenkombination $\boldsymbol{v} = (v_1, \ldots, v_\ell)$, die aufgrund der Domäne Dom von $\mathscr{L}^f$ möglich ist, genau eine Formel $\Psi(\boldsymbol{v})$ enthält.*
Für eine Formelmenge $\mathcal{S} \subset \mathscr{L}^f$ kann die Expansion durch

$$\mathrm{ex}(\mathcal{S}) := \bigcup_{\Psi(\boldsymbol{X}) \in \mathcal{S}} \mathrm{ex}(\Psi(\boldsymbol{X}))$$

berechnet werden. Dadurch kann die Formelmenge $\mathcal{S} \subset \mathscr{L}^f$ in eine Formelmenge $\mathrm{ex}(\mathcal{S}) \subset \mathscr{L}$ transformiert werden.

Durch das Konzept der Expansion kann eine endlich große Menge potentieller Geheimnisse $pot\_sec(\mathcal{C})$, die gemäß Definition 5.8 gebildet ist und Formeln aus $\mathscr{L}^f$ enthält, in eine unendlich große Formelmenge $\mathrm{ex}(pot\_sec(\mathcal{C}))$ transformiert werden, die ausschließlich Formeln aus $\mathscr{L}$ enthält. Dabei existiert in $\mathrm{ex}(pot\_sec(\mathcal{C}))$ für jede Formel $\Psi(\boldsymbol{X}) \in pot\_sec(\mathcal{C})$ und jede Konstantenkombination $\boldsymbol{v}$, die in $\Psi(\boldsymbol{X})$ für die freien Variablen aus $\boldsymbol{X}$ eingesetzt werden kann, eine Formel $\Psi(\boldsymbol{v})$, in der die freien Variablen aus $\boldsymbol{X}$ entsprechend der Konstantenkombination $\boldsymbol{v}$ ersetzt sind. Damit enthält $\Psi(\boldsymbol{v})$ anstelle der freien Variablen aus $\boldsymbol{X}$ Konstantenzeichen und entspricht damit einer Formel aus $\mathscr{L}$, auf die die Konzepte der DB-Interpretation und der DB-Implikation angewendet werden können.

Für die in Abbildung 5.2 gegebene Relationeninstanz *person* zu Relationenschema $\langle Person|A_{\mathrm{Person}}| \quad \rangle$ werden in Kapitel 5.2.2 die Vertraulichkeits-Constraints

- $c_0 = \{\texttt{Name}, \texttt{Geburtstag}\}$
- $c_1 = \{\texttt{Name}, \texttt{Plz}\}$

beispielhaft als Menge $\mathcal{C}$ von Vertraulichkeits-Constraints definiert. Wenn die Variable X_N dem Attribut `Name`, die Variable X_G dem Attribut `Geburtstag` und die Variable X_P dem Attribut `Plz` entspricht, kann nach Definition 5.8 die Menge $pot\_sec(\mathcal{C})$ durch die potentiellen Geheimnisse

- $\Psi_0((X_N, X_G)) := (\exists X_P)\ Person(X_N, X_G, X_P)$ für Constraint c_0 und
- $\Psi_1((X_N, X_P)) := (\exists X_G)\ Person(X_N, X_G, X_P)$ für Constraint c_1

als Vertraulichkeitspolitik der kontrollierten Anfrageauswertung gebildet werden. Dabei sind in Ψ_0 die Variablen X_N und X_G und in Ψ_1 die Variablen X_N und X_P frei enthalten und können dementsprechend beliebig mit Konstantenzeichen der Menge Dom von $\mathscr{L}^f$ belegt werden. Da in Dom alle Konstantenzeichen, die auch in *person* vorkommen, enthalten sein müssen, ist in der Expansion $\mathrm{ex}(\Psi_0((X_N, X_G)))$ des potentiellen Geheimnisses $\Psi_0((X_N, X_G))$ beispielsweise auch die Formel

$$(\exists X_P)\ Person(\text{Hellmann}, 03.01.1981, X_P)$$

enthalten, die keine freien Variablen enthält und deshalb Element der Sprache $\mathscr{L}$ ist. Durch diese Formel wird gefordert, dass die Werte `Hellmann` für Attribut `Name` und `03.01.1981` für Attribut `Geburtstag` für einen potentiellen Angreifer nicht gemeinsam in Erfahrung zu bringen sein dürfen, falls die Kombination dieser beiden Werte in einem Tupel von *person* enthalten ist.

5.2.4 Inferenzsicherheit bei nicht vorhandenem Vorwissen

Aufgrund der bisherigen Überlegungen ist nun bekannt, wie eine Fragment-Instanz f_s, die im Zuge der Fragmentierung einer ursprünglichen Relationeninstanz r nach dem Verfahren „Fragmentierung und partielle lokale Verwaltung“ entstanden ist, in einem Framework der kontrollierten Anfrageauswertung dargestellt werden kann. Ebenso ist es auch möglich, die Menge $\mathcal{C}$ von Vertraulichkeits-Constraints, die für das r zugrunde liegende Relationenschema $\langle R|A_R| \quad \rangle$ definiert wurde, in diesem Framework der kontrollierten Anfrageauswertung als eine Menge $pot\_sec(\mathcal{C})$ von potentiellen Geheimnissen zu modellieren.

Innerhalb des logik-orientierten Frameworks der kontrollierten Anfrageauswertung sollen nun beweisbare Aussagen zur Inferenzsicherheit des verwendeten Fragmentierungs-Ansatzes getroffen werden. Das heißt, es soll untersucht werden, inwieweit die durch die Vertraulichkeits-Constraints aus $\mathcal{C}$ definierten Vertraulichkeitsanforderungen durch den Einsatz des betrachteten Fragmentierungs-Ansatzes auch unter Berücksichtigung möglicher Inferenzen eingehalten werden können.

Das Framework der kontrollierten Anfrageauswertung basiert aber nicht nur auf einer logik-orientierten Datenbankinstanz und einer Vertraulichkeitspolitik. Zusätzlich dazu wird auch das Vorwissen eines potentiellen Angreifers betrachtet und explizit als Menge von Formeln aus der verwendeten logischen Sprache formalisiert (siehe Kapitel 4.1.3). Im Rahmen des nachfolgenden Beweises zur Inferenzsicherheit wird davon ausgegangen, dass das Vorwissen eines potentiellen Angreifers durch eine leere Menge von Formeln gegeben ist. Wenn mit *prior* das in der prädikatenlogischen Sprache $\mathscr{L}$ formalisierte Vorwissen bezeichnet wird, gilt also $prior = \emptyset$.

Das bedeutet aber nicht, dass das Vorwissen eines potentiellen Angreifers im Rahmen der hier gewählten Modellierung komplett außer Acht gelassen wird: In Kapitel 5.1.4 wird einem potentiellen Angreifer weitreichendes Vorwissen über die Funktionsweise des verwendeten Fragmentierungs-Ansatzes sowie die Kenntnis des Relationenschemas der ursprünglichen Relationeninstanz r zugestanden. Dieses Vorwissen wird bei der Modellierung des Fragmentierungs-Ansatzes im Framework der kontrollierten Anfrageauswertung berücksichtigt und ist deshalb implizit in der gewählten Modellierung enthalten.

So wird eine Fragment-Instanz f_s beispielsweise im Kontext ihrer ursprünglichen Relationeninstanz r modelliert, um im Rahmen der Modellierung zu berücksichtigen, dass ein potentieller Angreifer weiß, zu welchen Attributen des r zugrunde liegenden Relationenschemas ihm Werte vorenthalten bleiben. Ebenso wird auch das Wissen eines potentiellen Angreifers über die Semantik definierter Vertraulichkeitsanforderungen bei der Modellierung der Vertraulichkeitspolitik berücksichtigt:

Es wird bei der Konstruktion der Vertraulichkeitspolitik darauf geachtet, dass ein potentieller Angreifer zu schützende Informations-Aspekte nicht unmittelbar aus der modellierten Vertraulichkeitspolitik auslesen kann, indem er durch sein Vorwissen über die Konstruktion der Vertraulichkeitspolitik in Kombination mit den für ihn einsehbaren Inhalten dieser Schlussfolgerungen ziehen kann, die die gewünschten Vertraulichkeitsanforderungen verletzen.

Im Folgenden sei eine ursprüngliche Relationeninstanz r zu Schema $\langle R|A_R|\ \rangle$ mit $A_R = \{a_1, \ldots, a_{|A_R|}\}$ gegeben. Diese Relationeninstanz r sei im Zuge des Fragmentierungs-Ansatzes „Fragmentierung und partielle lokale Verwaltung“ gemäß Definition 3.7 derart in die Fragment-Instanzen f_o und f_s zerlegt worden, dass die zu dem Relationenschema $\langle R|A_R|\ \rangle$ berechnete Fragmentierung

$$\mathcal{F} = \{\langle F_o|A_{F_o}|SC_{F_o}\rangle, \langle F_s|A_{F_s}|SC_{F_s}\rangle\},$$

zu der f_o und f_s gebildet wurden, bezüglich einer Menge $\mathcal{C}$ von Vertraulichkeits-Constraints zu $\langle R|A_R|\ \rangle$ korrekt nach Definition 3.8 ist. Zum Nachweis der Inferenzsicherheit des Fragmentierungs-Ansatzes muss gezeigt werden, dass für die logik-orientierte Modellierung db_{f_s} der Fragment-Instanz f_s (siehe Kapitel 5.2.2), die der Sichtweise eines potentiellen Angreifers auf r entspricht, die Eigenschaft

$$\forall\, \Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C})) :\ prior \cup db_{f_s} \not\models_{DB} \Psi(\boldsymbol{v}) \tag{5.1}$$

erfüllt wird. Dabei bezeichnet $pot\_sec(\mathcal{C})$ die Modellierung der gegebenen Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für $\langle R|A_R|\ \rangle$ nach Definition 5.8. Da die Inferenzsicherheit hier unter der Annahme $prior = \emptyset$ gezeigt werden soll, kann die in Formel 5.1 dargestellte Eigenschaft offensichtlich zu

$$\forall\, \Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C})) :\ db_{f_s} \not\models_{DB} \Psi(\boldsymbol{v}) \tag{5.2}$$

vereinfacht werden.

Theorem 5.1 (Inferenzsicherheit bei leerem Vorwissen) *Sei gemäß Definition 2.2 eine Relationeninstanz r zu einem Relationenschema $\langle R|A_R|\ \rangle$ nach Definition 2.1 gegeben. Zu $\langle R|A_R|\ \rangle$ sei gemäß Definition 3.7 eine Fragmentierung $\mathcal{F}$ berechnet worden, die nach Definition 3.8 korrekt bezüglich einer nach Definition 3.2 konstruierten Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für $\langle R|A_R|\ \rangle$ sei. Unter der Annahme $prior = \emptyset$ kann gezeigt werden, dass Formel 5.2 gilt, wenn*

- *db_{f_s} der in Kapitel 5.2.2 vorgestellten logik-orientierten Modellierung der Fragment-Instanz f_s entspricht und f_s wiederum gemäß Definition 3.7 zu dem Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle \in \mathcal{F}$ bezüglich r gebildet ist und*
- *$\mathrm{ex}(pot\_sec(\mathcal{C}))$ die nach Definition 5.9 konstruierte Expansion der Menge $pot\_sec(\mathcal{C})$ ist, die wiederum der logik-orientierten Modellierung der Menge $\mathcal{C}$ gemäß Definition 5.8 entspricht.*

Beweis. Bezüglich der DB-Implikation (siehe Definition 5.2) gilt für eine Menge $\mathcal{S}$ von Formeln aus $\mathscr{L}$ und eine beliebige Formel Φ aus $\mathscr{L}$ genau dann die Beziehung $\mathcal{S} \models_{DB} \Phi$, wenn durch jede DB-Interpretation $\mathcal{I}$, die alle Formeln aus $\mathcal{S}$ erfüllt, auch die Formel Φ erfüllt wird. Damit für $\mathcal{S}$ und Φ die negierte Beziehung $\mathcal{S} \not\models_{DB} \Phi$ gilt, muss also mindestens eine DB-Interpretation $\mathcal{I}$ existieren, die zwar alle Formeln aus $\mathcal{S}$, aber *nicht* die Formel Φ erfüllt. Bezogen auf den zu erbringenden Beweis für die Gültigkeit von Formel 5.2 ist also nachzuweisen, dass für jede Formel $\Psi(\boldsymbol{v})$ aus $\mathrm{ex}(pot\_sec(\mathcal{C}))$ mindestens eine DB-Interpretation $\mathcal{I}$ gefunden werden kann, für die sowohl $\mathcal{I} \models_M db_{f_s}$ als auch $\mathcal{I} \not\models_M \Psi(\boldsymbol{v})$ gilt.

Dazu werde eine beliebige Formel $\tilde{\Psi}(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ mit $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$ betrachtet, die in der Expansion eines potentiellen Geheimnisses $\tilde{\Psi}(\boldsymbol{X}) \in pot\_sec(\mathcal{C})$, in dem die Variablen $\boldsymbol{X} = (X_{i_1}, \ldots, X_{i_k})$ frei vorkommen, enthalten ist. Aufgrund der Überlegungen bezüglich der Anforderungen an eine DB-Interpretation zur Erfüllung eines potentiellen Geheimnisses (siehe Kapitel 5.2.3) ist klar, dass eine DB-Interpretation $\mathcal{I}$, die $\tilde{\Psi}(\boldsymbol{v})$ erfüllen soll, stets ein Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}(R)$ enthalten muss, in dem für jedes $j \in \{i_1, \ldots, i_k\}$ jeweils $u_j = v_j$ gilt.

Da in der Menge $pot\_sec(\mathcal{C})$ das potentielle Geheimnis $\tilde{\Psi}(\boldsymbol{X})$ mit den freien Variablen $\boldsymbol{X} = (X_{i_1}, \ldots, X_{i_k})$ enthalten ist, muss es aufgrund der Konstruktionsweise von $pot\_sec(\mathcal{C})$ nach Definition 5.8 ein Vertraulichkeits-Constraint $c = \{a_{i_1}, \ldots, a_{i_k}\}$ in $\mathcal{C}$ geben, wenn für jedes $j \in \{i_1, \ldots, i_k\}$ der Term t_j, der in der prädikatenlogischen Formulierung $\tilde{\Psi}(\boldsymbol{X})$ des Vertraulichkeits-Constraints c eine freie Variable X_j repräsentiert, jeweils dem Attribut a_j aus Schema $\langle R|A_R| \quad \rangle$ entspricht. Andernfalls wäre $\tilde{\Psi}(\boldsymbol{X})$ nicht in $pot\_sec(\mathcal{C})$ enthalten. Da die Fragmentierung $\mathcal{F}$ des Schemas $\langle R|A_R| \quad \rangle$ nach Voraussetzung korrekt bezüglich $\mathcal{C}$ ist (siehe Definition 3.8), muss für Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle \in \mathcal{F}$ die Bedingung $c \not\subseteq A_{F_s}$ erfüllt werden. Demnach existiert mindestens ein Attribut $a_m \in c$, für das $a_m \notin A_{F_s}$ gilt. Aufgrund dessen müssen *alle* Formeln aus $db_{f_s}^+ \subset db_{f_s}$, durch die gemäß Definition 5.6 positives Wissen der Fragment-Instanz f_s im Kontext von r modelliert wird, die Form

$$\ldots (\exists X_m) \ldots \; R(t_1, \ldots, t_{|A_R|})$$

haben, wobei $t_m := X_m$ gilt. Dabei ist entscheidend, dass der dem Attribut a_m entsprechende Term t_m in *jeder* Formel aus $db_{f_s}^+$ ausschließlich einer existenzquantifizierten Variablen X_m entsprechen kann.

Zu db_{f_s} soll im Folgenden eine DB-Interpretation $\mathcal{I}^*$ konstruiert werden, für die sowohl $\mathcal{I}^* \models_M db_{f_s}$ als auch $\mathcal{I}^* \not\models_M \tilde{\Psi}(\boldsymbol{v})$ gilt. Dazu wird $\mathcal{I}^*$ derart konstruiert, dass für jede Formel Φ aus $db_{f_s}^+ \subset db_{f_s}$ jeweils genau ein Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ enthalten ist. Für dieses Tupel müssen im Folgenden die Werte seiner einzelnen Komponenten $u_1, \ldots, u_{|A_R|}$ definiert werden. Im Rahmen dessen wird für die Komponente u_j der Wert des Terms t_j der Formel Φ gewählt, falls der Term t_j in Φ einem

Konstantenzeichen entspricht. Dabei gilt hier stets $j \neq m$, weil t_m nach Konstruktion in *jeder* Formel aus $db^+_{f_s}$ einer existenzquantifizierten Variablen entspricht. Nach den Überlegungen aus Kapitel 5.2.2 zur Erfüllbarkeit von Formeln aus $db^+_{f_s}$ ist damit bereits sichergestellt, dass durch die hier zu konstruierende DB-Interpretation $\mathcal{I}^*$ alle Formeln aus der Menge $db^+_{f_s}$ erfüllt werden können, obwohl für jedes Tupel aus $\mathcal{I}^*(R)$ bislang jeweils nur ein Teil seiner Komponenten definiert ist.

Hinsichtlich der Erfüllbarkeit von Φ kann der Wert für eine Komponente u_h des betrachteten Tupels $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ hingegen frei aus dem Universum $\mathcal{U}$ von $\mathcal{I}^*$ gewählt werden, falls der Term t_h in Φ eine existenzquantifizierte Variable X_h repräsentiert. Nach den Überlegungen aus Kapitel 5.2.2 wird die Frage der Erfüllbarkeit einer Formel aus $db^+_{f_s}$ durch eine DB-Interpretation allein anhand der Terme dieser Formel, die Konstantenzeichen repräsentieren, entschieden. Dabei wird $\mathcal{I}^*$ hier derart konstruiert, dass der Wert für die betrachtete Komponente u_h des Tupels $(u_1, \ldots, u_{|A_R|})$ komplett frei aus dem Universum $\mathcal{U}$ gewählt wird, falls $h \neq m$ gilt. Für die Komponente u_m wird der Wert aus $\mathcal{U} \setminus \{v_m\}$ gewählt, wobei v_m aus der Konstantenkombination $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$ stammt. Eine solche Wahl für u_m ist dabei stets möglich, da nach Definition einer DB-Interpretation das Universum $\mathcal{U}$ einer DB-Interpretation stets der Menge *Dom* der Konstantenzeichen der zugrunde liegenden prädikatenlogischen Sprache $\mathscr{L}$ entsprechen muss. Da *Dom* (und damit auch $\mathcal{U}$) nach Voraussetzung unendlich groß ist, gilt stets $|\mathcal{U} \setminus \{v_m\}| \geq 1$ und die gewünschte Wahl $u_m \in \mathcal{U} \setminus \{v_m\}$ ist damit stets möglich.

Wenn $\mathcal{I}^*$ keine weiteren Tupel als die für $db^+_{f_s}$ konstruierten enthält, wird nach den Überlegungen aus Kapitel 5.2.2 zur Erfüllbarkeit des Completeness-Sentence auch $db^-_{f_s}$ erfüllt: Nach diesen Überlegungen muss der zu Fragment-Instanz f_s konstruierte Completeness-Sentence aufgrund seiner ausnahmslos allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ unter *allen* möglichen Konstantenersetzungen dieser Variablen durch die DB-Interpretation $\mathcal{I}^*$ erfüllt werden, damit dieser durch $\mathcal{I}^*$ erfüllt wird. Unter einer Konstantenersetzung $(X_1/u'_1), \ldots, (X_{|A_R|}/u'_{|A_R|})$ kann das Disjunkt $\neg R(X_1, \ldots, X_{|A_R|})$ des konstruierten Completeness-Sentence durch $\mathcal{I}^*$ erfüllt werden, wenn das Tupel $(u'_1, \ldots, u'_{|A_R|}) \in \mathcal{U}^{|A_R|}$ *nicht* in $\mathcal{I}^*(R)$ enthalten ist. Dabei ist es ausreichend, dass dieses eine Disjunkt des Completeness-Sentence unter der betrachteten Konstantenersetzung durch $\mathcal{I}^*$ erfüllt wird, damit der komplette Completeness-Sentence unter dieser Konstantenersetzung durch $\mathcal{I}^*$ erfüllt wird.

Für ein Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ existiert unter der getroffenen Annahme, dass $\mathcal{I}^*$ keine weiteren Tupel als die für $db^+_{f_s}$ konstruierten enthält, gemäß der Konstruktion von $\mathcal{I}^*(R)$ stets eine Formel Φ in $db^+_{f_s}$, in der für jeden Term t_j von Φ, der einem Konstantenzeichen entspricht, jeweils $t_j = u_j$ gilt. Das bedeutet gemäß der Konstruktion von Φ (siehe Definition 5.6), dass auch in der Fragment-

Instanz f_s ein Tupel μ existieren muss, das jedem Attribut $a_j \in A_{F_s} \cap A_R$ den Wert u_j aus $(u_1, \ldots, u_{|A_R|})$ zuweist. Der zu f_s nach Definition 5.7 konstruierte Completeness-Sentence muss aufgrund dieses Tupels $\mu \in f_s$ ein Disjunkt in Form einer Konjunktion über die Attribute aus $A_{F_s} \cap A_R$ enthalten. Dieses Disjunkt kann – unabhängig von der Konstruktion von $\mathcal{I}^*$ durch die Semantik des ausgezeichneten Prädikatensymbols = einer DB-Interpretation – erfüllt werden, wenn für die allquantifizierten Variablen des Completeness-Sentence die Konstantenersetzung $(X_1/u_1), \ldots, (X_{|A_R|}/u_{|A_R|})$ betrachtet wird. Auch hier ist es ausreichend, dass dieses eine Disjunkt des Completeness-Sentence unter der betrachteten Konstantenersetzung durch $\mathcal{I}^*$ erfüllt wird, damit der komplette Completeness-Sentence unter dieser Konstantenersetzung durch $\mathcal{I}^*$ erfüllt wird.

Damit werden also sowohl $db_{f_s}^+$ als auch $db_{f_s}^-$ durch die wie oben vorgestellt konstruierte DB-Interpretation $\mathcal{I}^*$ erfüllt. Als Folge dessen wird auch die Formelmenge db_{f_s} durch $\mathcal{I}^*$ erfüllt. Die Formel $\tilde{\Psi}(\boldsymbol{v})$ wird hingegen wie gefordert *nicht* durch $\mathcal{I}^*$ erfüllt: Nach Konstruktion von $\mathcal{I}^*$ existiert *kein* Tupel $(u_1, \ldots, u_{|A_R|})$ in $\mathcal{I}^*(R)$, in dem hinsichtlich der Komponente u_m die Eigenschaft $u_m = v_m$ gilt. In $\tilde{\Psi}(\boldsymbol{v})$ repräsentiert der Term t_m aber nach Voraussetzung das Konstantenzeichen v_m aus $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$. Es existiert also *kein* Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$, in dem bezüglich $\boldsymbol{v}$ für jedes $j \in \{i_1, \ldots, i_k\}$ jeweils $u_j = v_j$ gilt. Die Gültigkeit dieser Eigenschaft ist aber zwingende Voraussetzung dafür, dass die DB-Interpretation $\mathcal{I}^*$ die Formel $\tilde{\Psi}(\boldsymbol{v})$ erfüllen kann. □

Damit ist nachgewiesen, dass ein potentieller Angreifer unter der vereinbarten Annahme $prior = \emptyset$ einen geschützten Informations-Aspekt, der durch eine Formel $\Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ beschrieben wird, allein aufgrund seiner Kenntnis von db_{f_s} nicht erschließen kann. Der oben vorgestellte Beweis zeigt, dass für jeden zu schützenden Informations-Aspekt $\Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ stets eine DB-Interpretation $\mathcal{I}$ konstruiert werden kann, die zwar db_{f_s}, aber nicht $\Psi(\boldsymbol{v})$ erfüllt.

In Kapitel 5.1.1 wird erläutert, dass durch eine DB-Interpretation $\mathcal{I}$ stets eine vollständige Datenbankinstanz induziert wird. Da die verwendete prädikatenlogische Sprache $\mathscr{L}$, die dem verwendeten Framework der kontrollierten Anfrageauswertung zugrunde liegt, zur logik-orientierten Modellierung der ursprünglichen Relationeninstanz r dient, wird hier durch eine DB-Interpretation $\mathcal{I}$ für $\mathscr{L}$, die db_{f_s} erfüllt, also stets eine Relationeninstanz r' induziert, die aus Sicht eines potentiellen Angreifers der ursprünglichen Relationeninstanz r entsprechen könnte.

Wie in dem oben vorgestellten Beweis zur Inferenzsicherheit gezeigt wird, kann für jeden zu schützenden Informations-Aspekt $\Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ immer mindestens eine DB-Interpretation $\mathcal{I}$ gefunden werden, unter der zwar db_{f_s}, aber nicht $\Psi(\boldsymbol{v})$ erfüllt wird. Das bedeutet, dass aus der Sicht eines potentiellen Angreifers,

die durch db_{f_s} beschrieben wird, stets die Existenz einer durch $\mathcal{I}$ induzierten Relationeninstanz r' möglich sein muss, in der ein bestimmter geschützter Informations-Aspekt nicht gilt. Im Falle eines geschützten Informations-Aspekts, der in der ursprünglichen Relationeninstanz r gültig ist, handelt es sich bei r' – analog zu den vorgestellten Ansätzen der kontrollierten Anfrageauswertung aus Kapitel 4 – um eine alternative mögliche Relationeninstanz, die aus Sicht des potentiellen Angreifers ununterscheidbar zu r ist.

5.2.5 Inferenzsicherheit bei vorhandenem Vorwissen

Aufgrund der Ergebnisse aus Kapitel 5.2.4 ist nun bekannt, dass der dort untersuchte Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ sicher vor unerwünschten Inferenzen ist, sofern sichergestellt ist, dass einem potentiellen Angreifer kein explizites Vorwissen zur Verfügung steht. Mit explizitem Vorwissen wird dabei alles das (in *prior* formalisierte) Wissen bezeichnet, das einem potentiellen Angreifer über die Inhalte der ursprünglichen Relationeninstanz oder das verwendete System bekannt ist und nicht bereits explizit oder implizit bei der Modellierung der Sichtweise eines solchen potentiellen Angreifers berücksichtigt wird. Dabei ist die für das Vorwissen vereinbarte Einschränkung, dass $prior = \emptyset$ gilt, von essentieller Bedeutung für die Gültigkeit der Eigenschaft der Inferenzsicherheit. Im Folgenden soll an einem Beispiel verdeutlicht werden, dass die Eigenschaft der Inferenzsicherheit nicht ohne Weiteres gegeben ist, wenn von dieser Einschränkung des Vorwissens abgesehen wird.

Dass von Einschränkungen des Vorwissens eines Benutzers aber im Sinne der Inferenzsicherheit nicht vollständig abgesehen werden kann, ist unmittelbar einzusehen: Wenn ein zu schützender Informations-Aspekt $\Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ einem potentiellen Angreifer bereits als explizites Vorwissen zur Verfügung steht (es gilt also $\Psi(\boldsymbol{v}) \in prior$), kann durch Fragmentierungs-Ansätze – oder auch sonstige Ansätze zur Wahrung von Vertraulichkeitsanforderungen – nicht verhindert werden, dass dieser potentielle Angreifer Kenntnis von $\Psi(\boldsymbol{v})$ hat, da einem Benutzer Wissen, das diesem bereits bekannt ist, nicht wieder entzogen werden kann (siehe Kapitel 1.1.1). Diese Eigenschaft, dass einmal erworbenes Wissen zwar erweitert, aber nicht revidiert werden kann, wird im Kontext logik-orientierter Frameworks auch als Monotonie bezeichnet [5, Kap. 7.2].

Um die oben beschriebene Situation auszuschließen, wird für das Vorwissen eines Benutzers stets die Einschränkung

$$\forall\, \Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C})) :\ prior \not\models_{DB} \Psi(\boldsymbol{v}) \tag{5.3}$$

gefordert [18, Abschnitt 2]. Für die Erfüllbarkeit dieser Eigenschaft ist dabei zwingende Voraussetzung, dass die Formelmenge *prior* in sich konsistent sein muss. Andernfalls gäbe es *keine* DB-Interpretation, die *prior* erfüllt. Damit würde trivialerweise durch jede DB-Interpretation, die *prior* erfüllt, auch eine beliebige Formel $\Phi \in \mathscr{L}$ erfüllt. Gemäß der Definition der DB-Implikation würden also insbesondere auch alle zu schützenden Informations-Aspekte aus $\mathrm{ex}(pot\_sec(\mathcal{C}))$ unmittelbar durch eine inkonsistente Formelmenge *prior* impliziert. Diese hier diskutierte Forderung aus Formel 5.3 ist in Kapitel 5.2.4 für $prior = \emptyset$ trivialerweise erfüllt.

Aber auch bei gegebenem Vorwissen in Form einer Menge *prior*, die die in Formel 5.3 geforderte Eigenschaft erfüllt, ist Inferenzsicherheit nicht zwangsläufig gegeben. Um dies zu demonstrieren, soll an dieser Stelle wieder die Fragmentierung der Relationeninstanz *person* betrachtet werden, die aus Abbildung 5.3 bekannt ist. Die prädikatenlogische Modellierung des positiven Wissens, das in der zu *person* gebildeten Fragment-Instanz f_s bezogen auf *person* enthalten ist, ist (wie aus Kapitel 5.2.2 bekannt) durch die Formelmenge

$$\begin{aligned} db^+_{f_s} = \{ \quad & (\exists X_N)\ Person(\ X_N, \quad 03.01.1981, \quad 94142\ ), \\ & (\exists X_N)\ Person(\ X_N, \quad 12.02.1952, \quad 94139\ ), \\ & (\exists X_N)\ Person(\ X_N, \quad 03.01.1981, \quad 94139\ ) \quad \} \end{aligned}$$

gegeben. Dabei entspricht diese Formelmenge $db^+_{f_s}$ wie gehabt der Sichtweise, die ein potentieller Angreifer auf Basis seiner Kenntnis der Fragment-Instanz f_s auf das in *person* enthaltene positive Wissen hat.

Angenommen ein potentieller Angreifer weiß, dass die Person `Hellmann`, über die unter Umständen Wissen in der Relationeninstanz *person* enthalten sein könnte, in einem kleinen Ort mit der Postleitzahl `94142` wohnt und in diesem Ort die einzige Person ist, die am `03.01.1981` Geburtstag hat. Da einem potentiellen Angreifer nach Voraussetzung die Attributmenge des *person* zugrunde liegenden Relationenschemas bekannt ist, kann diese Information aus der Sichtweise eines potentiellen Angreifers in *prior* durch die Formel

$$(\exists X_N)\ Person(X_N, 03.01.1981, 94142) \Rightarrow Person(\mathrm{Hellmann}, 03.01.1981, 94142)$$

der Sprache $\mathscr{L}$ ausgedrückt werden. Dabei ist für *prior* die in Formel 5.3 geforderte Eigenschaft erfüllt, falls *prior* keine weiteren Formeln außer der oben genannten enthält: Da in *prior* die Prämisse dieser Formel nicht erfüllt wird, muss auch die Konklusion dieser Formel nicht notwendigerweise gelten, so dass durch *prior* insbesondere auch kein geschützter Informations-Aspekt impliziert werden kann.

Wenn der betrachtete potentielle Angreifer *prior* und $db_{f_s}^+$ zusammen betrachtet, wird aber durch $prior \cup db_{f_s}^+$ unmittelbar die Formel

$$Person(\text{Hellmann}, 03.01.1981, 94142)$$

impliziert. Wenn also wie in dem Beispiel aus Kapitel 5.2.3 ein potentielles Geheimnis $\Psi_0((X_N, X_G))$ in $pot\_sec(C)$ existiert, in dessen Expansion $\text{ex}(\Psi_0((X_N, X_G)))$ die Formel

$$(\exists X_P)\ Person(\text{Hellmann}, 03.01.1981, X_P)$$

enthalten ist, wird durch die betrachtete Kombination aus *prior* und $db_{f_s}^+$ die durch $pot\_sec(C)$ gegebene Vertraulichkeitspolitik verletzt, obwohl durch *prior* die in Formel 5.3 geforderte Einschränkung erfüllt wird. Daraus ergibt sich, dass diese Einschränkung für *prior* nicht stark genug ist, um unter dieser Einschränkung die Eigenschaft der Inferenzsicherheit nachweisen zu können.

5.2.6 Inferenzsicherheit unter funktionalen Abhängigkeiten

Aus den Überlegungen in Kapitel 5.2.5 ergibt sich, dass der Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ im Allgemeinen nicht inferenzsicher ist, wenn einem potentiellen Angreifer Vorwissen zur Verfügung steht. Dies gilt auch unter der Bedingung, dass es nicht möglich sein darf, allein aus dem betrachteten Vorwissen heraus einen vertraulichen Informations-Aspekt herleiten zu können (vgl. Formel 5.3). Auf der anderen Seite ist aber aus Kapitel 5.2.4 bekannt, dass unter der Annahme $prior = \emptyset$, welche die stärkste denkbare Einschränkung des Vorwissens darstellt, die Inferenzsicherheit dieses Fragmentierungs-Ansatzes formal nachgewiesen werden kann. Ziel muss es also möglichst sein, bestimmte Einschränkungen für das Vorwissen eines potentiellen Angreifers zu finden, die zwar weniger restriktiv als $prior = \emptyset$ sind, aber unter denen dennoch die Inferenzsicherheit des betrachteten Fragmentierungs-Ansatzes nachgewiesen werden kann. Ein solches Resultat soll im Folgenden für Mengen funktionaler Abhängigkeiten, die einer bestimmten Restriktion unterliegen, gezeigt werden.

Wie aus Kapitel 2.2 bekannt ist, werden funktionale Abhängigkeiten auf der Ebene des Relationenschemas definiert. Da einem potentiellen Angreifer das Relationenschema, das einer ursprünglichen Relationeninstanz zugrunde liegt, nach Voraussetzung bekannt ist (siehe Kapitel 5.1.4), sind diesem auch die funktionalen Abhängigkeiten, die für dieses Relationenschema vereinbart sind, bekannt und können als Vorwissen eines potentiellen Angreifers aufgefasst werden (siehe auch [10, Abschnitt 5]). Des Weiteren wird in [15, Abschnitt 1] demonstriert, dass funktionale Abhängigkeiten prinzipiell als Grundlage für Inferenzen, durch die vertrauliche

Information erschlossen werden kann, dienen können. Deshalb ist es nicht selbstverständlich, dass für den Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ die Eigenschaft der Inferenzsicherheit unter (eingeschränkten) funktionalen Abhängigkeiten nachgewiesen werden kann.

Im Rahmen des nachfolgenden Beweises werden dabei für das Relationenschema $\langle R|A_R|SC_R\rangle$, das einer betrachteten ursprünglichen Relationeninstanz r zugrunde liegt, ausschließlich funktionale Abhängigkeiten der Form $A \to \{a\}$ untersucht, für die $A \subseteq A_R$ und $a \in A_R$ gilt. Dass dabei keine funktionalen Abhängigkeiten der aus Definition 2.5 bekannten allgemeinen Form untersucht werden, nach der auch Abhängigkeiten der Form $A \to \{a_{i_1}, \ldots, a_{i_k}\}$ mit $A \subseteq A_R$, $\{a_{i_1}, \ldots, a_{i_k}\} \subseteq A_R$ und $k > 1$ existieren können, ist dabei *keine* Einschränkung der Mächtigkeit funktionaler Abhängigkeiten: Eine solche funktionale Abhängigkeit kann stets in eine Menge

$$A \to \{a_{i_1}\}, A \to \{a_{i_2}\}, \ldots, A \to \{a_{i_k}\}$$

von funktionalen Abhängigkeiten überführt werden, die offensichtlich semantisch äquivalent zu $A \to \{a_{i_1}, \ldots, a_{i_k}\}$ ist.

Um funktionale Abhängigkeiten auch in dem entwickelten Framework der kontrollierten Anfrageauswertung als Vorwissen berücksichtigen zu können, müssen diese in eine geeignete logik-orientierte Darstellung überführt werden. Dies kann in Anlehnung an [7, Kap. 15.1] folgendermaßen geschehen:

Definition 5.10 (Funktionale Abhängigkeiten im Vorwissen) *Sei das Relationenschema einer ursprünglichen Relationeninstanz r gemäß Definition 2.1 durch $\langle R|A_R|SC_R\rangle$ mit $A_R = \{a_1, \ldots, a_{|A_R|}\}$ gegeben. Wenn in SC_R eine Menge Σ von funktionalen Abhängigkeiten gemäß Definition 2.5 enthalten ist, die ausnahmslos die Form $A \to \{a\}$ haben, kann eine funktionale Abhängigkeit*

$$\{a_{e_1}, \ldots, a_{e_\ell}\} \to \{a_e\}$$

aus Σ mit $\{a_{e_1}, \ldots, a_{e_\ell}\} \subseteq A_R$ und $a_e \in A_R$ durch die Formel

$$\begin{aligned}(\forall X_1)\ldots(\forall X_{|A_R|})(\forall Y_1)\ldots(\forall Y_{|A_R|})[\ & (R(X_1, \ldots, X_{|A_R|}) \wedge R(Y_1, \ldots, Y_{|A_R|}) \wedge \\ & X_{e_1} = Y_{e_1} \wedge \ldots \wedge X_{e_\ell} = Y_{e_\ell}) \\ & \Rightarrow X_e = Y_e \qquad\qquad]\end{aligned}$$

in der prädikatenlogischen Sprache $\mathscr{L}$ modelliert werden. Dabei sind $X_1, \ldots, X_{|A_R|}$ und $Y_1, \ldots, Y_{|A_R|}$ Variablen aus der Menge Var der Sprache $\mathscr{L}$.
Die funktionalen Abhängigkeiten aus Σ können damit im Vorwissen eines potentiellen Angreifers durch eine Menge $prior_\Sigma$ von Formeln aus $\mathscr{L}$ modelliert werden, in der für jede funktionale Abhängigkeit $\{a_{e_1}, \ldots, a_{e_\ell}\} \to \{a_e\}$ aus Σ jeweils genau eine Formel der oben vorgestellten Form enthalten ist.

Nach den Überlegungen aus Kapitel 5.1.1 muss eine Formel $\Gamma \in prior_\Sigma$ aufgrund der ausnahmslos allquantifizierten Variablen $X_1, \ldots, X_{|A_R|}$ und $Y_1, \ldots, Y_{|A_R|}$ durch eine DB-Interpretation $\mathcal{I}$ unter *jeder* möglichen Konstantenersetzung dieser Variablen erfüllt werden, damit Γ durch $\mathcal{I}$ erfüllt wird. Dabei muss bezogen auf die Erfüllbarkeit von Γ durch $\mathcal{I}$ unter einer konkreten Konstantenersetzung der allquantifizierten Variablen grundsätzlich zwischen zwei Fällen unterschieden werden: Falls unter der betrachteten Konstantenersetzung die Prämisse von Γ *nicht* durch $\mathcal{I}$ erfüllt wird, muss unter dieser Konstantenersetzung auch die Konklusion von Γ *nicht* zwangsläufig durch $\mathcal{I}$ erfüllt werden, um Γ durch $\mathcal{I}$ erfüllen zu können. Falls unter der betrachteten Konstantenersetzung hingegen die Prämisse von Γ durch $\mathcal{I}$ erfüllt wird, muss unter dieser Konstantenersetzung auch die Konklusion von Γ zwingend durch $\mathcal{I}$ erfüllt werden, damit Γ durch die DB-Interpretation $\mathcal{I}$ erfüllt werden kann.

Im Folgenden sei (analog zu den Annahmen zu Theorem 5.1) eine ursprüngliche Relationeninstanz r zu Schema $\langle R|A_R|SC_R\rangle$ mit $A_R = \{a_1, \ldots, a_{|A_R|}\}$ gegeben. Diese Relationeninstanz r sei im Zuge des Fragmentierungs-Ansatzes „Fragmentierung und partielle lokale Verwaltung“ gemäß Definition 3.7 derart in die Fragment-Instanzen f_o und f_s zerlegt worden, dass die zu dem Relationenschema $\langle R|A_R|SC_R\rangle$ berechnete Fragmentierung

$$\mathcal{F} = \{\langle F_o|A_{F_o}|SC_{F_o}\rangle, \langle F_s|A_{F_s}|SC_{F_s}\rangle\},$$

zu der f_o und f_s gebildet wurden, bezüglich einer Menge $\mathcal{C}$ von Vertraulichkeits-Constraints zu $\langle R|A_R|SC_R\rangle$ korrekt nach Definition 3.8 ist. Bezogen auf die lokalen semantischen Bedingungen, die in SC_R vereinbart sind, wird davon ausgegangen, dass in SC_R ausschließlich eine Menge Σ von funktionalen Abhängigkeiten der Form $A \rightarrow \{a\}$ enthalten ist. Dabei unterliegt Σ der Restriktion, dass eine funktionale Abhängigkeit $A \rightarrow \{a\}$ aus Σ *nicht* derart gestaltet sein darf, dass für diese die beiden Eigenschaften $A \not\subseteq A_{F_s} \cap A_R$ und $a \in A_{F_s} \cap A_R$ erfüllt sind. Das heißt in Σ darf *keine* funktionale Abhängigkeit $A \rightarrow \{a\}$ existieren, in der das Attribut $a \in A_R$ in der Attributmenge A_{F_s} enthalten ist und mindestens ein Attribut aus $A \subseteq A_R$ nicht in A_{F_s} liegt.

Innerhalb der prädikatenlogischen Modellierung im Framework der kontrollierten Anfrageauswertung kann eine solche Menge Σ von funktionalen Abhängigkeiten, die einem potentiellen Angreifer nach Voraussetzung bekannt ist, wie in Definition 5.10 beschrieben durch eine Menge $prior_\Sigma$ von Formeln aus $\mathscr{L}$ formuliert werden. Unter der Annahme, dass einem potentiellen Angreifer kein weiteres explizites Vorwissen als $prior_\Sigma$ zur Verfügung steht, muss zum Nachweis der Inferenzsicherheit des Fragmentierungs-Ansatzes gezeigt werden, dass die Eigenschaft

$$\forall\, \Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C})) :\ prior_\Sigma \cup db_{f_s} \not\models_{DB} \Psi(\boldsymbol{v}) \tag{5.4}$$

erfüllt wird, wenn db_{f_s} die logik-orientierte Modellierung der Fragment-Instanz f_s bezeichnet (siehe Kapitel 5.2.2), die der Sichtweise eines potentiellen Angreifers auf r entspricht. Des Weiteren bezeichnet $pot\_sec(\mathcal{C})$ die Modellierung der gegebenen Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für $\langle R|A_R|SC_R\rangle$ nach Definition 5.8.

Theorem 5.2 (Inferenzsicherheit unter funktionalen Abh.) *Sei gemäß Definition 2.2 eine Relationeninstanz r zu einem Relationenschema $\langle R|A_R|SC_R\rangle$ nach Definition 2.1 gegeben. Zu $\langle R|A_R|SC_R\rangle$ sei gemäß Definition 3.7 eine Fragmentierung $\mathcal{F}$ berechnet worden, die nach Definition 3.8 korrekt bezüglich einer nach Definition 3.2 konstruierten Menge $\mathcal{C}$ von Vertraulichkeits-Constraints für $\langle R|A_R|SC_R\rangle$ sei. Es kann gezeigt werden, dass Formel 5.4 gilt, wenn*

- *db_{f_s} der in Kapitel 5.2.2 vorgestellten logik-orientierten Modellierung der Fragment-Instanz f_s entspricht und f_s wiederum gemäß Definition 3.7 zu dem Fragment $\langle F_s|A_{F_s}|SC_{F_s}\rangle \in \mathcal{F}$ bezüglich r gebildet ist,*
- *$\mathrm{ex}(pot\_sec(\mathcal{C}))$ die nach Definition 5.9 konstruierte Expansion der Menge $pot\_sec(\mathcal{C})$ ist, die wiederum der logik-orientierten Modellierung der Menge $\mathcal{C}$ gemäß Definition 5.8 entspricht, und*
- *$prior_\Sigma$ gemäß Definition 5.10 für die Menge $\Sigma = SC_R$ konstruiert ist, in der ausschließlich funktionale Abhängigkeiten $A \rightarrow \{a\}$ nach Definition 2.5 enthalten sind, für die nicht $A \not\subseteq A_{F_s} \cap A_R$ und $a \in A_{F_s} \cap A_R$ gilt.*

Beweis. Bezogen auf den zu erbringenden Beweis für die Gültigkeit von Formel 5.4 ist analog zu dem Beweis von Theorem 5.1 zu zeigen, dass für jede Formel $\Psi(\boldsymbol{v})$ aus $\mathrm{ex}(pot\_sec(\mathcal{C}))$ mindestens eine DB-Interpretation $\mathcal{I}$ gefunden werden kann, für die jeweils $\mathcal{I} \models_M db_{f_s}$, $\mathcal{I} \models_M prior_\Sigma$ und $\mathcal{I} \not\models_M \Psi(\boldsymbol{v})$ gilt. Dazu werde auch hier eine beliebige Formel $\tilde{\Psi}(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ mit $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$ betrachtet, die in der Expansion eines potentiellen Geheimnisses $\tilde{\Psi}(\boldsymbol{X}) \in pot\_sec(\mathcal{C})$, in dem die Variablen $\boldsymbol{X} = (X_{i_1}, \ldots, X_{i_k})$ frei vorkommen, enthalten ist. Aufgrund der Existenz von $\tilde{\Psi}(\boldsymbol{X})$ in $pot\_sec(\mathcal{C})$ kann analog zu dem Beweis von Theorem 5.1 argumentiert werden, dass *alle* Formeln aus $db^+_{f_s} \subset db_{f_s}$ die Form

$$\ldots(\exists X_m)\ldots\ R(t_1, \ldots, t_{|A_R|})$$

mit $t_m := X_m$ haben, so dass für einen Index $m \in \{i_1, \ldots, i_k\}$ der Term t_m in *jeder* Formel aus $db^+_{f_s}$ einer existenzquantifizierten Variablen X_m entspricht.

Im Folgenden enthalte die Indexmenge $\mathrm{Ind}^+_{F_s}$ genau die Indizes $j \in \{1, \ldots, |A_R|\}$, für die der Term t_j in einer beliebigen Formel $\Phi \in db^+_{f_s}$ ein Konstantenzeichen repräsentiert. Es gilt also insbesondere $m \notin \mathrm{Ind}^+_{F_s}$. Die Menge $prior_\Sigma$, die alle funktionalen

Abhängigkeiten aus Σ in der logik-orientierten Darstellung nach Definition 5.10 enthält, wird nun in die drei Mengen $prior^1_\Sigma$, $prior^2_\Sigma$ und $prior^3_\Sigma$ aufgeteilt. Dabei erfolgt diese Aufteilung so, dass eine beliebige funktionale Abhängigkeit

$$(\forall X_1)\dots(\forall X_{|A_R|})(\forall Y_1)\dots(\forall Y_{|A_R|})[\ (R(X_1,\dots,X_{|A_R|}) \wedge R(Y_1,\dots,Y_{|A_R|}) \wedge X_{e_1} = Y_{e_1} \wedge \dots \wedge X_{e_\ell} = Y_{e_\ell}) \Rightarrow X_e = Y_e\ ]$$

aus der Menge $prior_\Sigma$ jeweils genau dann der Menge

- $prior^1_\Sigma$ zugeteilt wird, wenn $\{e_1,\dots,e_\ell\} \subseteq \mathrm{Ind}^+_{F_s}$ und $e \in \mathrm{Ind}^+_{F_s}$ gilt,
- $prior^2_\Sigma$ zugeteilt wird, wenn $\{e_1,\dots,e_\ell\} \subseteq \mathrm{Ind}^+_{F_s}$ und $e \notin \mathrm{Ind}^+_{F_s}$ gilt, und
- $prior^3_\Sigma$ zugeteilt wird, wenn $\{e_1,\dots,e_\ell\} \not\subseteq \mathrm{Ind}^+_{F_s}$ und $e \notin \mathrm{Ind}^+_{F_s}$ gilt.

Der Fall, dass für eine Formel aus $prior_\Sigma$ der oben dargestellten Form sowohl $\{e_1,\dots,e_\ell\} \not\subseteq \mathrm{Ind}^+_{F_s}$ als auch $e \in \mathrm{Ind}^+_{F_s}$ gilt, kann hier nicht eintreten, da die Menge Σ, auf der $prior_\Sigma$ basiert, nach Voraussetzung derart eingeschränkt ist, dass in Σ ausschließlich funktionale Abhängigkeiten der Form $A \to \{a\}$ enthalten sind, für die nicht $A \not\subseteq A_{F_s} \cap A_R$ und $a \in A_{F_s} \cap A_R$ gilt.

Im Folgenden soll eine DB-Interpretation $\mathcal{I}^*$ konstruiert werden, für die jeweils die Eigenschaften $\mathcal{I}^* \models_M db_{f_s}$, $\mathcal{I}^* \models_M prior_\Sigma$ und $\mathcal{I}^* \not\models_M \tilde{\Psi}(\boldsymbol{v})$ gelten. Dazu wird $\mathcal{I}^*$ hier derart konstruiert, dass für jede Formel Φ aus $db^+_{f_s} \subset db_{f_s}$ jeweils genau ein Tupel $(u_1,\dots,u_{|A_R|}) \in \mathcal{I}^*(R)$ enthalten ist, in dem für alle $j \in \mathrm{Ind}^+_{F_s}$ für die Komponente u_j der Wert des Terms t_j der Formel Φ gewählt wird. Die Komponenten u_h des Tupels $(u_1,\dots,u_{|A_R|})$ mit $h \notin \mathrm{Ind}^+_{F_s}$ bleiben vorerst undefiniert. Nach den Überlegungen aus Kapitel 5.2.2 zur Erfüllbarkeit von Formeln aus $db^+_{f_s}$ ist damit trotz der noch undefinierten Komponenten der Tupel aus $\mathcal{I}^*(R)$ bereits sichergestellt, dass durch die hier zu konstruierende DB-Interpretation $\mathcal{I}^*$ alle Formeln aus der Menge $db^+_{f_s}$ erfüllt werden können.

In $prior^1_\Sigma$ sind nach Konstruktion genau die Formeln enthalten, die funktionalen Abhängigkeiten der Form $A \to \{a\}$ mit $A \subseteq A_{F_s} \cap A_R$ und $a \in A_{F_s} \cap A_R$ aus Σ entsprechen. Diese funktionalen Abhängigkeiten sind für das Relationenschema $\langle R|A_R|SC_R\rangle$, das der ursprünglichen Relationeninstanz r zugrunde liegt, vereinbart und müssen damit in r erfüllt werden. Die Fragment-Instanz f_s entspricht – abgesehen von den in f_s enthaltenen Tupel-IDs – einer Projektion der ursprünglichen Relationeninstanz r auf die Attributmenge $A_{F_s} \cap A_R$, aus der auch alle Attribute stammen, die in den betrachteten funktionalen Abhängigkeiten der Form $A \to \{a\}$ mit $A \subseteq A_{F_s} \cap A_R$ und $a \in A_{F_s} \cap A_R$ aus Σ vorkommen. Da die logik-orientierte Modellierung $db^+_{f_s}$ jede Wertekombination aus einem Tupel der Projektion von r

auf $A_{F_s} \cap A_R$ durch eine Formel repräsentiert, muss – aufgrund der Gültigkeit der betrachteten funktionalen Abhängigkeiten aus Σ in dieser Projektion – durch eine beliebige DB-Interpretation $\mathcal{I}$, die $db^+_{f_s}$ erfüllt, auch jede Formel aus $prior^1_\Sigma$ erfüllt werden. Deshalb gilt auch $\mathcal{I}^* \models_M prior^1_\Sigma$.

Obwohl durch die hier zu konstruierende DB-Interpretation $\mathcal{I}^*$ bereits die Formelmengen $db^+_{f_s}$ und $prior^1_\Sigma$ erfüllt werden, sind noch nicht alle Komponenten der Tupel aus $\mathcal{I}^*(R)$ definiert. Analog zu dem Beweis von Theorem 5.1 kann auch hier für jede Komponente u_h eines Tupels $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ mit $h \notin \mathrm{Ind}^+_{F_s}$ ein beliebiger Wert aus dem $\mathcal{I}^*$ zugrunde liegenden Universum $\mathcal{U}$ gewählt werden, ohne dass dabei die Erfüllbarkeit von $db^+_{f_s}$ oder $prior^1_\Sigma$ durch $\mathcal{I}^*$ beeinträchtigt werden kann. Für alle Indizes

$$h \in \{1, \ldots, |A_R|\} \setminus (\mathrm{Ind}^+_{F_s} \cup \{m\})$$

wird dazu ein beliebiger Wert aus dem Universum $\mathcal{U}$ gewählt, der in *jedem* Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ der jeweiligen Komponente u_h dieses Tupels zugewiesen wird. Für den Index $m \notin \mathrm{Ind}^+_{F_s}$ wird ein beliebiger Wert aus der Menge $\mathcal{U} \setminus \{v_m\}$ gewählt, der ebenfalls in *jedem* Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$ der jeweiligen Komponente u_m dieses Tupels zugewiesen wird. Dabei stammt v_m aus der Konstantenkombination $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$, die in $\tilde{\Psi}(\boldsymbol{v})$ enthalten ist. Wie in dem Beweis zu Theorem 5.1 erläutert wird, ist eine solche Wahl für u_m aufgrund des nach Annahme unendlich großen Universums $\mathcal{U}$ stets möglich.

Analog zu den entsprechenden Überlegungen aus dem Beweis zu Theorem 5.1 wird auch hier der zu f_s gebildete Completeness-Sentence durch die in diesem Beweis konstruierte DB-Interpretation $\mathcal{I}^*$ erfüllt, wenn $\mathcal{I}^*$ keine weiteren Tupel als die für $db^+_{f_s}$ konstruierten enthält. Deshalb wird auch hier $db^-_{f_s}$ durch $\mathcal{I}^*$ erfüllt, so dass insgesamt $\mathcal{I}^* \models_M db_{f_s}$ gilt. Es bleibt damit noch zu zeigen, dass durch $\mathcal{I}^*$ auch $prior^2_\Sigma$ und $prior^3_\Sigma$ erfüllt werden, damit auch die Eigenschaft $\mathcal{I}^* \models_M prior_\Sigma$ für die konstruierte DB-Interpretation $\mathcal{I}^*$ nachgewiesen ist.

Um zu zeigen, dass $\mathcal{I}^* \models_M prior^2_\Sigma$ gilt, muss – wie im Vorfeld dieses Beweises erläutert wird – gezeigt werden, dass die Konklusion jeder Formel $\Gamma \in prior^2_\Sigma$ unter jeder möglichen Konstantenersetzung der allquantifizierten Variablen durch $\mathcal{I}^*$ erfüllt wird, unter der auch die Prämisse von Γ durch $\mathcal{I}^*$ erfüllt wird. Dazu werde im Folgenden eine beliebige durch Γ bezeichnete Formel

$$\begin{aligned}(\forall X_1) \ldots (\forall X_{|A_R|})(\forall Y_1) \ldots (\forall Y_{|A_R|})[\ & (R(X_1, \ldots, X_{|A_R|}) \wedge R(Y_1, \ldots, Y_{|A_R|}) \wedge \\ & X_{e_1} = Y_{e_1} \wedge \ldots \wedge X_{e_\ell} = Y_{e_\ell}) \\ & \Rightarrow X_e = Y_e \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad]\end{aligned}$$

aus $prior^2_\Sigma$ unter einer beliebigen möglichen Konstantenersetzung

$$(X_1/u_1), \ldots, (X_{|A_R|}/u_{|A_R|})(Y_1/w_1), \ldots, (Y_{|A_R|}/w_{|A_R|})$$

betrachtet. Falls unter dieser Konstantenersetzung die Prämisse von Γ durch $\mathcal{I}^*$ erfüllt wird, müssen in $\mathcal{I}^*(R)$ die beiden Tupel $(u_1, \ldots, u_{|A_R|})$ und $(w_1, \ldots, w_{|A_R|})$ enthalten sein. Damit unter dieser Konstantenersetzung auch die Konklusion von Γ erfüllt wird, muss für die beiden betrachteten Tupel aus $\mathcal{I}^*(R)$ die Eigenschaft $u_e = w_e$ gelten. Diese Eigenschaft ist erfüllt, da nach Konstruktion von $\mathcal{I}^*$ für alle $h \in \{1, \ldots, |A_R|\} \setminus \mathrm{Ind}^+_{F_s}$ für zwei beliebige Tupel $(u'_1, \ldots, u'_{|A_R|}) \in \mathcal{I}^*(R)$ und $(w'_1, \ldots, w'_{|A_R|}) \in \mathcal{I}^*(R)$ stets die Eigenschaft $u'_h = w'_h$ gilt und des Weiteren für die Konklusion der betrachteten Formel $\Gamma \in prior^2_\Sigma$ nach Konstruktion von $prior^2_\Sigma$ die Eigenschaft $e \notin \mathrm{Ind}^+_{F_s}$ gelten muss.

Der Beweis für die Gültigkeit von $\mathcal{I}^* \models_M prior^3_\Sigma$ verläuft offensichtlich analog zu dem oben vorgestellten Beweis für $\mathcal{I}^* \models_M prior^2_\Sigma$, so dass damit neben $\mathcal{I}^* \models_M db_{f_s}$ auch die geforderte Eigenschaft $\mathcal{I}^* \models_M prior_\Sigma$ gezeigt ist. Es bleibt also noch die Gültigkeit von $\mathcal{I}^* \not\models_M \tilde{\Psi}(\boldsymbol{v})$ zu zeigen: Nach Konstruktion von $\mathcal{I}^*$ existiert *kein* Tupel $(u_1, \ldots, u_{|A_R|})$ in $\mathcal{I}^*(R)$, in dem hinsichtlich der Komponente u_m die Eigenschaft $u_m = v_m$ gilt. In $\tilde{\Psi}(\boldsymbol{v})$ repräsentiert der Term t_m aber nach Voraussetzung das Konstantenzeichen v_m aus $\boldsymbol{v} = (v_{i_1}, \ldots, v_{i_k})$. Es existiert also *kein* Tupel $(u_1, \ldots, u_{|A_R|}) \in \mathcal{I}^*(R)$, in dem bezüglich $\boldsymbol{v}$ für jedes $j \in \{i_1, \ldots, i_k\}$ jeweils $u_j = v_j$ gilt. Die Gültigkeit dieser Eigenschaft ist aber zwingende Voraussetzung dafür, dass die DB-Interpretation $\mathcal{I}^*$ die Formel $\tilde{\Psi}(\boldsymbol{v})$ erfüllen kann. □

Damit ist nachgewiesen, dass ein potentieller Angreifer unter den Voraussetzungen aus Theorem 5.2 einen geschützten Informations-Aspekt, der durch eine Formel $\Psi(\boldsymbol{v}) \in \mathrm{ex}(pot\_sec(\mathcal{C}))$ beschrieben wird, aufgrund seiner Kenntnis von db_{f_s} und $prior_\Sigma$ nicht erschließen kann. Analog zu Kapitel 5.2.4 kann überlegt werden, dass damit aus der Sicht eines potentiellen Angreifers für jeden zu schützenden Informations-Aspekt stets die Existenz einer (alternativen) Relationeninstanz möglich sein muss, die aus Sicht dieses Angreifers ununterscheidbar zu der ursprünglichen Relationeninstanz ist und in der dieser Informations-Aspekt nicht gültig ist.

6 Evaluation und Ausblick

Motiviert durch den Stellenwert von Information in der heutigen Informationsgesellschaft wird in Kapitel 1 dieser Diplomarbeit die Bedeutung von Mechanismen zur Durchsetzung von Vertraulichkeitsanforderungen in Informationssystemen dargelegt. In diesem Kontext wird zwischen Mechanismen zur Zugriffskontrolle, die ausschließlich den Zugriff auf einzelne Daten eines Systems regulieren, und Mechanismen zur Inferenzkontrolle, die auch den Informationsgehalt von Daten und mögliche Inferenzen eines Benutzers berücksichtigen, differenziert.

Durch die Fragmentierungs-Ansätze aus Kapitel 3 soll vor allem der Schutz sensibler Assoziationen zwischen Informations-Aspekten ermöglicht werden. Dazu wird eine Relationeninstanz mit Hilfe von Projektionen in verschiedene Fragment-Instanzen zerlegt. Für dieses Vorgehen existieren verschiedene Ansätze, die in Kapitel 3 dieser Diplomarbeit in einer vereinheitlichten Notation vorgestellt werden. Dabei wird im Rahmen der Erläuterung dieser Verfahren – in Anlehnung an die in Kapitel 2.1 vorgestellte Notation für das relationale Datenmodell – stets zwischen der Schema- und der Instanz-Ebene eines Datenbanksystems differenziert.

Für diese Fragmentierungs-Ansätze wird von den jeweiligen Autoren aber jeweils nicht erläutert, ob diese auch Schutz vor (unerwünschten) Inferenzen bieten, durch die möglicherweise sensible Information erschlossen werden kann. Im Gegensatz dazu ist für die Verfahren der kontrollierten Anfrageauswertung, deren Grundlagen in Kapitel 4 vorgestellt werden, die Eigenschaft der Inferenzsicherheit formal nachgewiesen. Ziel ist es deshalb, in dieser Diplomarbeit beweisbare Aussagen zur Inferenzsicherheit der Fragmentierungs-Ansätze zu ermöglichen, indem diese Ansätze in dem logik-orientierten Framework der kontrollierten Anfrageauswertung modelliert werden. Deshalb werden in Kapitel 5.1 dieser Diplomarbeit konkrete Ausprägungen einzelner Komponenten dieses Frameworks spezifiziert, die in dieser Kombination ein logik-orientiertes Framework bilden, in dem eine Modellierung der Fragmentierungs-Ansätze aus Kapitel 3 möglich ist.

Darauf aufbauend werden dann in Kapitel 5.2 konkrete Untersuchungen zur Inferenzsicherheit des in Kapitel 3.4 vorgestellten Fragmentierungs-Ansatzes „Fragmentierung und partielle lokale Verwaltung“ vorgenommen. Bei diesem Ansatz

wird eine Relationeninstanz in zwei Fragment-Instanzen zerlegt und lediglich eine (bestimmte) dieser beiden Fragment-Instanzen ist einem potentiellen Angreifer bekannt. Um die Untersuchung der Inferenzsicherheit dieses Fragmentierungs-Ansatzes zu ermöglichen, wird in Kapitel 5.2.2 die Sichtweise, die ein potentieller Angreifer aufgrund der ihm bekannten Fragment-Instanz auf die ursprüngliche Relationeninstanz hat, in dem entwickelten Framework der kontrollierten Anfrageauswertung modelliert. Ebenso müssen auch die Vertraulichkeitsanforderungen, die bei den Fragmentierungs-Ansätzen durch Vertraulichkeits-Constraints definiert werden, in dem Framework der kontrollierten Anfrageauswertung durch eine geeignete Vertraulichkeitspolitik modelliert werden (siehe Kapitel 5.2.3).

Zum Nachweis der Inferenzsicherheit des betrachteten Fragmentierungs-Ansatzes ist zu zeigen, dass ein potentieller Angreifer auf Basis der für ihn modellierten Sichtweise auf die ursprüngliche Relationeninstanz und auch seines Vorwissens keine Möglichkeit hat, einen beliebigen vertraulichen Informations-Aspekt rational zu erschließen. Dabei stellt sich als ein wesentliches Resultat heraus, dass die Frage, ob der betrachtete Fragmentierungs-Ansatz inferenzsicher ist, abhängig von der Art des konkret betrachteten Vorwissens eines potentiellen Angreifers ist. Für das Vorwissen eines potentiellen Angreifers werden aber von den Autoren, die die in Kapitel 3 erwähnten Fragmentierungs-Ansätze vorstellen, jeweils keine expliziten Annahmen getroffen. Deshalb muss hier unter eigenen Annahmen für das vorhandene Vorwissen gearbeitet werden.

Konkret wird in Kapitel 5.2.4 dieser Diplomarbeit nachgewiesen, dass der betrachtete Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ inferenzsicher ist, sofern davon ausgegangen werden kann, dass einem potentiellen Angreifer kein explizites Vorwissen – das heißt Vorwissen, das nicht bereits implizit in der logik-orientierten Modellierung berücksichtigt wird – zur Verfügung steht. Diese Annahme, dass ein potentieller Angreifer über kein explizites Vorwissen verfügt, dürfte sich in der Praxis aber oft als unzutreffend erweisen.

Aufgrund dessen wird in Kapitel 5.2.5 untersucht, ob die Eigenschaft der Inferenzsicherheit auch unter beliebigem Vorwissen nachgewiesen werden kann. Dabei stellt sich heraus, dass dies – auch unter der Voraussetzung, dass es nicht möglich ist, allein durch das Vorwissen eines potentiellen Angreifers einen vertraulichen Informations-Aspekt herleiten zu können – nicht der Fall ist. Dass es aber prinzipiell möglich ist, Inferenzsicherheit auch unter geeignet eingeschränktem Vorwissen zu erzielen und nachweisen zu können, wird in Kapitel 5.2.6 gezeigt: Dort werden Mengen funktionaler Abhängigkeiten, die einer bestimmten Restriktion unterliegen, als Vorwissen eines potentiellen Angreifers betrachtet.

Motiviert durch dieses Resultat kann es das Ziel zukünftiger Forschungsarbeiten sein, weitere (möglichst schwache) Restriktionen für das Vorwissen eines poten-

tiellen Angreifers zu finden, unter denen für den Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ die Eigenschaft der Inferenzsicherheit nachgewiesen werden kann. Im Rahmen dessen kann beispielsweise der Nachweis versucht werden, dass für dieses Fragmentierungs-Verfahren die Eigenschaft der Inferenzsicherheit auch unter Mengen funktionaler Abhängigkeiten gilt, die nicht der in Kapitel 5.2.6 vorausgesetzten Restriktion unterliegen.

Eine weiterer Ansatzpunkt, geeignete Restriktionen für das Vorwissen eines potentiellen Angreifers zu finden, könnte in Anlehnung an [15] in der Suche nach geeigneten allgemeinen (syntaktischen) Einschränkungen für die logische Sprache, in der das Vorwissen eines Benutzers ausgedrückt werden kann, liegen. In [15] gelingt es, Inferenzsicherheit durch (im Allgemeinen nicht inferenzsichere) diskretionäre Zugriffskontrolle zu erzielen, indem unter anderem Einschränkungen der Anfragesprache und der Sprachen zur Definition der Vertraulichkeitspolitik und des Vorwissens eines Benutzers vorausgesetzt werden.

Dabei kann hier im Rahmen der logik-orientierten Modellierung des betrachteten Fragmentierungs-Ansatzes aber lediglich die Sprache für das Vorwissen eingeschränkt werden: Eine Einschränkung der Anfragesprache ist nicht ohne Weiteres möglich, weil einem potentiellen Angreifer nach Voraussetzung die komplette Fragment-Instanz f_s bekannt ist, die für ihn zugänglich gespeichert ist, während er auf die lokal gespeicherte Fragment-Instanz f_o überhaupt keinen Zugriff bekommen kann. Ebenso kommt eine Einschränkung der Sprache zur Definition der Vertraulichkeitspolitik nicht ohne Weiteres in Frage, da die zu formulierende Vertraulichkeitspolitik durch die in Kapitel 5.2.3 gewählte logik-orientierte Modellierung der Vertraulichkeits-Constraints vorgegeben ist.

Von den drei Fragmentierungs-Ansätzen, die in Kapitel 3 vorgestellt werden, werden in dieser Diplomarbeit lediglich für einen dieser drei Ansätze Untersuchungen zur Inferenzsicherheit durchgeführt. In zukünftigen Forschungsarbeiten könnte deshalb auch versucht werden, Ergebnisse zur Inferenzsicherheit der beiden anderen in Kapitel 3 vorgestellten Fragmentierungs-Ansätze zu erzielen. Dazu kann analog zu Kapitel 5.2 versucht werden, eine geeignete logik-orientierte Modellierung dieser Ansätze zu entwickeln. Derartige Modellierungen können dabei auch auf den grundsätzlichen Überlegungen zur Entwicklung eines Frameworks der kontrollierten Anfrageauswertung aus Kapitel 5.1 aufbauen: Die dort beschriebenen Komponenten eines solchen Frameworks sind so gewählt, dass auf dieser Basis eine logik-orientierte Modellierung aller Ansätze aus Kapitel 3 möglich sein sollte.

Im Rahmen dessen dürfte sich der Fragmentierungs-Ansatz „Fragmentierung und nicht kooperierende Partner“ aus Kapitel 3.2 weitestgehend analog zu den Ideen aus Kapitel 5.2 modellieren lassen: Auch bei diesem Ansatz existieren zwei Fragment-Instanzen und ein potentieller Angreifer hat nach Voraussetzung genau auf eine

(beliebige) dieser beiden Fragment-Instanzen Zugriff. Dabei muss aber im Rahmen der Modellierung geeignet mit verschlüsselten Attributwerten umgegangen werden können. Falls davon ausgegangen werden kann, dass ein potentieller Angreifer außer der Existenz dieser Werte überhaupt kein Wissen über diese in Erfahrung bringen kann, könnten diese unter Umständen – in Anlehnung an die Modellierung der Attributwerte der Fragment-Instanz f_o in Kapitel 5.2.2 – durch existenzquantifizierte Variablen formalisiert werden.

Bei der Modellierung des Fragmentierungs-Ansatzes „Fragmentierung und partielle Verschlüsselung“ aus Kapitel 3.3 muss neben der Existenz verschlüsselter Attributwerte auch beachtet werden, dass ein potentieller Angreifer im Rahmen dieses Ansatzes Kenntnis von allen existierenden Fragment-Instanzen erlangen kann. Dabei ist vor allem von Bedeutung, dass die Kombination des Wissens aus verschiedenen Fragment-Instanzen nicht als Grundlage für Inferenzen, die die Vertraulichkeitsanforderungen verletzen, dienen kann. Resultate zur Inferenzsicherheit dieses Ansatzes können aber darüber hinaus auch von Interesse sein, um die in Kapitel 3.3.1 nicht näher begründete Annahme, dass ein nicht autorisierter Betrachter einzelne Fragment-Instanzen ausschließlich über den natürlichen Verbund korrekt zusammensetzen kann (siehe Fußnote 6 auf Seite 32), entweder verifizieren oder falsifizieren zu können.

Die in Kapitel 3 vorgestellten Fragmentierungs-Ansätze und die in dieser Diplomarbeit erzielten Ergebnisse zur Inferenzsicherheit – sowie eventuell zukünftige Ergebnisse diesbezüglich – könnten auch für den in [3] und [33] vorgestellten Ansatz von Interesse sein. Wie bei dem in Kapitel 5.2 untersuchten Fragmentierungs-Ansatz „Fragmentierung und partielle lokale Verwaltung“ ist es das erklärte Ziel, eine Datenbankinstanz durch vertikale Fragmentierung in zwei Fragment-Instanzen zu zerlegen, von denen eine sicher vor unbefugtem Zugriff verwaltet wird. Allerdings wird in diesen Ausarbeitungen schwerpunktmäßig die effiziente Beantwortung von Anfragen eines Benutzers behandelt. Eine gewünschte Fragmentierung muss im Rahmen dieses Ansatzes explizit angegeben werden und kann nicht – wie bei den in Kapitel 3 vorgestellten Fragmentierungs-Ansätzen – durch ein abstrahierendes Regelwerk (wie beispielsweise durch Vertraulichkeits-Constraints) ausgedrückt werden, auf dessen Basis dann eine geeignete Fragmentierung berechnet werden kann. Die Suche nach derartigen Verfahren wird in [3, Abschnitt 7] als ein möglicher Ansatzpunkt für weitere Forschungsarbeiten vorgeschlagen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Abiteboul, Serge, Richard Hull und Victor Vianu: *Foundations of Databases*. Addison-Wesley, Reading, 1995, ISBN 0-201-53771-0.

[2] Aggarwal, Gagan, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas und Ying Xu: *Two Can Keep A Secret: A Distributed Architecture for Secure Database Services*. In: *2nd Biennial Conference on Innovative Data Systems Research, CIDR 2005*, Seiten 186–199, 2005.

[3] Anciaux, Nicolas, Mehdi Benzine, Luc Bouganim, Philippe Pucheral und Dennis Shasha: *GhostDB: Querying Visible and Hidden Data Without Leaks*. In: Chan, Chee Yong, Beng Chin Ooi und Aoying Zhou (Herausgeber): *ACM International Conference on Management of Data, SIGMOD 2007*, Seiten 677–688. ACM, 2007, ISBN 978-1-59593-686-8.

[4] Anderson, Ross J.: *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, Indianapolis, 2. Auflage, 2008, ISBN 978-0-470-06852-6.

[5] Beierle, Christoph und Gabriele Kern-Isberner: *Methoden wissensbasierter Systeme – Grundlagen, Algorithmen, Anwendungen*. Vieweg + Teubner, Wiesbaden, 4. Auflage, 2008, ISBN 978-3-8348-0504-1.

[6] Bishop, Matt: *Computer Security: Art and Science*. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.

[7] Biskup, Joachim: *Grundlagen von Informationssystemen*. Vieweg-Lehrbuch Informatik. Vieweg, Braunschweig Wiesbaden, 1995, ISBN 3-528-05494-8.

[8] Biskup, Joachim: *For unknown secrecies refusal is better than lying*. Data & Knowledge Engineering, 33(1):1–23, 2000.

[9] Biskup, Joachim: *Security in Computing Systems – Challenges, Approaches and Solutions*. Springer, Berlin Heidelberg, 2009, ISBN 978-3-540-78441-8.

[10] Biskup, Joachim: *Usability Confinement of Server Reactions: Maintaining Inference-Proof Client Views by Controlled Interaction Execution.* In: Kikuchi, Shinji, Shelly Sachdeva und Subhash Bhalla (Herausgeber): *Databases in Networked Information Systems, DNIS 2010*, Band 5999 der Reihe *Lecture Notes in Computer Science*, Seiten 80–106. Springer, 2010, ISBN 978-3-642-12037-4.

[11] Biskup, Joachim und Piero A. Bonatti: *Lying versus refusal for known potential secrets.* Data & Knowledge Engineering, 38(2):199–222, 2001.

[12] Biskup, Joachim und Piero A. Bonatti: *Controlled query evaluation for enforcing confidentiality in complete information systems.* International Journal of Information Security, 3(1):14–27, 2004.

[13] Biskup, Joachim und Piero A. Bonatti: *Controlled query evaluation for known policies by combining lying and refusal.* Annals of Mathematics and Artificial Intelligence, 40(1–2):37–62, 2004.

[14] Biskup, Joachim und Piero A. Bonatti: *Controlled query evaluation with open queries for a decidable relational submodel.* Annals of Mathematics and Artificial Intelligence, 50(1–2):39–77, 2007.

[15] Biskup, Joachim, David W. Embley und Jan-Hendrik Lochner: *Reducing inference control to access control for normalized database schemas.* Information Processing Letters, 106(1):8–12, 2008.

[16] Biskup, Joachim, Jan-Hendrik Lochner und Sebastian Sonntag: *Optimization of the Controlled Evaluation of Closed Relational Queries.* In: Gritzalis, Dimitris und Javier Lopez (Herausgeber): *Emerging Challenges for Security, Privacy and Trust, IFIP 2009*, Band 297 der Reihe *Advances in Information and Communication Technology*, Seiten 214–225. Springer, 2009, ISBN 978-3-642-01243-3.

[17] Biskup, Joachim und Torben Weibert: *Keeping secrets in incomplete databases.* International Journal of Information Security, 7(3):199–217, 2008.

[18] Biskup, Joachim und Lena Wiese: *Preprocessing for controlled query evaluation with availability policy.* Journal of Computer Security, 16(4):477–494, 2008.

[19] Biskup, Joachim und Lena Wiese: *Combining Consistency and Confidentiality Requirements in First-Order Databases.* In: Samarati, Pierangela, Moti Yung, Fabio Martinelli und Claudio Agostino Ardagna (Herausgeber): *12th International Conference on Information Security, ISC 2009*, Band 5735 der Reihe *Lecture Notes in Computer Science*, Seiten 121–134. Springer, 2009, ISBN 978-3-642-04473-1.

[20] Bonatti, Piero A., Sarit Kraus und V. S. Subrahmanian: *Foundations of Secure Deductive Databases.* IEEE Transactions on Knowledge and Data Engineering, 7(3):406–422, 1995.

[21] Castano, Silvana, Maria Grazia Fugini, Giancarlo Martella und Pierangela Samarati: *Database Security.* ACM Press books. Addison-Wesley & ACM Press, New York, 1995, ISBN 0-201-59375-0.

[22] Ceselli, Alberto, Ernesto Damiani, Sabrina De Capitani di Vimercati, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Modeling and Assessing Inference Exposure in Encrypted Databases.* ACM Transactions on Information and System Security, 8(1):119–152, 2005.

[23] Ciriani, Valentina, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Fragmentation and Encryption to Enforce Privacy in Data Storage.* In: Biskup, Joachim und Javier Lopez (Herausgeber): *12th European Symposium on Research in Computer Security, ESORICS 2007*, Band 4734 der Reihe *Lecture Notes in Computer Science*, Seiten 171–186. Springer, 2007, ISBN 978-3-540-74834-2.

[24] Ciriani, Valentina, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Enforcing Confidentiality Constraints on Sensitive Databases with Lightweight Trusted Clients.* In: Gudes, Ehud und Jaideep Vaidya (Herausgeber): *Data and Applications Security XXIII, DBSec 2009*, Band 5645 der Reihe *Lecture Notes in Computer Science*, Seiten 225–239. Springer, 2009, ISBN 978-3-642-03006-2.

[25] Ciriani, Valentina, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases.* In: *29th IEEE International Conference on Distributed Computing Systems, ICDCS 2009*, Seiten 32–39. IEEE Computer Society, 2009, ISBN 978-0769536590.

[26] Ciriani, Valentina, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Keep a Few: Outsourcing Data While Maintaining Confidentiality.* In: Backes, Michael und Peng Ning (Herausgeber): *14th European Symposium on Research in Computer Security, ESORICS 2009*, Band 5789 der Reihe *Lecture Notes in Computer Science*, Seiten 440–455. Springer, 2009, ISBN 978-3-642-04443-4.

[27] Ciriani, Valentina, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi und Pierangela Samarati: *Combining Fragmentation and Encryption to Protect Privacy in Data Storage.* ACM Transactions on Information and System Security, TISSEC, 13(3), 2010.

[28] Eckert, Claudia: *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag, München, 4. Auflage, 2006, ISBN 978-3-486-57851-5.

[29] Farkas, Csilla und Sushil Jajodia: *The Inference Problem: A Survey*. ACM SIGKDD Explorations Newsletter, 4(2):6–11, 2002.

[30] Gollmann, Dieter: *Computer Security*. John Wiley and Sons, Chichester, 2. Auflage, 2006, ISBN 978-0-470-86293-3.

[31] Hacigümüs, Hakan, Sharad Mehrotra und Balakrishna R. Iyer: *Providing Database as a Service*. In: *Proceedings of the 18th International Conference on Data Engineering, ICDE 2002*, Seiten 29–40. IEEE Computer Society, 2002, ISBN 0-7695-1531-2.

[32] Nerode, Anil und Richard A. Shore: *Logic for Applications*. Graduate texts in computer science. Springer, New York, 2. Auflage, 1997, ISBN 0-387-94893-7.

[33] Salperwyck, Christophe, Nicolas Anciaux, Mehdi Benzine, Luc Bouganim, Philippe Pucheral und Dennis Shasha: *GhostDB: Hiding Data from Prying Eyes*. In: Koch, Christoph, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl Christian Kanne, Wolfgang Klas und Erich J. Neuhold (Herausgeber): *33rd International Conference on Very Large Data Bases, VLDB 2007*, Seiten 1346–1349. ACM, 2007, ISBN 978-1-59593-649-3.

[34] Samarati, Pierangela und Sabrina De Capitani di Vimercati: *Data Protection in Outsourcing Scenarios: Issues and Directions*. In: Feng, Dengguo, David A. Basin und Peng Liu (Herausgeber): *ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010*, Seiten 1–14. ACM, 2010, ISBN 978-1-60558-936-7.

[35] Schneier, Bruce: *Applied Cryptography – Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, New York, 2. Auflage, 1996, ISBN 0-471-11709-9.

[36] Sicherman, George L., Wiebren de Jonge und Reind P. van de Riet: *Answering Queries Without Revealing Secrets*. ACM Transactions on Database Systems, 8(1):41–59, 1983.

[37] Sweeney, Latanya: *k-Anonymity: A Model for Protecting Privacy*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5):557–570, 2002.

[38] Özsu, M. Tamer und Patrick Valduriez: *Principles of Distributed Database Systems*. Prentice Hall, Upper Saddle River, 2. Auflage, 1999, ISBN 0-13-659707-6.

Selbstständigkeitserklärung

Hiermit versichere ich, Marcel Preuß, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, 30. September 2010

Einverständniserklärung des Urhebers

Ich, Marcel Preuß, erkläre mich einverstanden, dass diese von mir verfasste Diplomarbeit nach §6 (1) des UrhG der Öffentlichkeit durch Übernahme in die Bereichsbibliothek zugänglich gemacht wird. Damit können Leser der Bibliothek die Arbeit einsehen und zu persönlichen wissenschaftlichen Zwecken Kopien aus dieser Arbeit anfertigen. Weitere Urheberrechte werden nicht berührt.

Dortmund, 30. September 2010