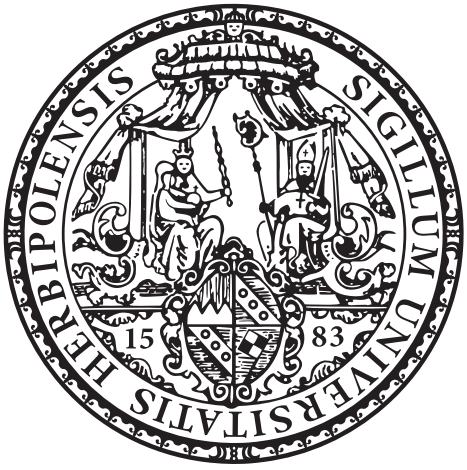

Universität Würzburg

Masterarbeit

Vergleich und Integration der Data Warehouse Architekturen PaDaWaN und i2b2

Verfasser:
Leon Liman

Betreuer:
Prof. Dr. Frank Puppe
Dipl.-Inf. Georg Fette

Datum der Abgabe:

31. März 2017

Inhaltsverzeichnis

1 Einleitung

In medizinischen Einrichtungen fallen Tag für Tag große Mengen an unterschiedlichsten Daten an, die dann in der Regel in irgendeiner Form gespeichert werden. Da dies oft in verschiedensten Formaten und Systemen geschieht, besteht in vielen dieser Einrichtungen ein Interesse daran, Auswertungen über sämtliche Daten mit einem zentralen System durchführen zu können. Ein solches wird dann als klinisches „Data Warehouse“ (vergleiche [1]) bezeichnet. Es ermöglicht die Auswertung aller vorhandener Daten durch sowohl einzelne Mitarbeiter, die sich beispielsweise die Krankengeschichte eines konkreten Patienten ansehen wollen, als auch zu Forschungszwecken, wobei zum Beispiel Zusammenhänge in den Daten gesucht werden. Ein weiterer Einsatzbereich wären Verwaltungsaufgaben innerhalb einzelner Abteilungen oder auch einer kompletten medizinischen Einrichtung, mit beispielsweise dem Ziel der Optimierung interner Abläufe.
Damit dies funktioniert, lassen sich viele klinische Data Warehouse Systeme in zwei sehr grundlegende Komponenten unterteilen:

- Zunächst gibt es eine **Terminologie** (im Folgenden auch als *Ontology* und *Katalog* bezeichnet), die festlegt, welche Arten von Daten überhaupt in dem System gespeichert werden können. Dies können zum Beispiel mögliche Laborwerte, verschiedene Diagnosen oder auch demographische Daten von Patienten sein.
- Der zweite Bestandteil sind dann die eigentlichen **medizinischen Daten**, also die Werte, die für konkrete Patienten im System gespeichert werden. Diese Daten werden weiter unten auch als *Fakten* sowie als *Informationen* bezeichnet. Jeder Wert wird immer einem Eintrag aus der *Terminologie* zugeordnet und in einem einheitlichen Format im Data Warehouse abgelegt.

Ohne so ein zentrales System zur Datenverwaltung müssten für die oben beschriebenen (und auch für andere mögliche) Einsatzzwecke manuell die benötigten Daten zusammengesucht und ausgewertet werden. Somit bieten klinische Data Warehouse Systeme in vielen Bereichen auch eine deutliche Zeitersparnis.

Die folgende Arbeit betrachtet die beiden Data Warehouse Architekturen PaDaWaN [2] und i2b2 [3]. Beide Systeme basieren auf den zwei oben beschriebenen Komponenten und ermöglichen die dort beschriebenen Einsatzzwecke. Der PaDaWaN wird bereits am Universitätsklinikum Würzburg eingesetzt, während i2b2 eine von verschiedenen medizinischen Einrichtungen eingesetzte Open Source Lösung ist.

1.1 Aufgabenstellung

Diese Arbeit setzt sich konkret aus den folgenden beiden Aspekten zusammen.

1.1.1 Beschreibung und Vergleich der Systeme

Zunächst sollen beide Architekturen miteinander verglichen werden. Zu diesem Zweck werden die Systeme als Erstes jeweils einzeln beschrieben. Als Teil dieser Beschreibung wird die jeweils verwendete Datenstruktur genauer erläutert. Außerdem werden die Möglichkeiten zur Nutzung der in den Systemen vorhandenen Daten über Anfragen und graphische Benutzeroberflächen vorgestellt. Anschließend werden die unterschiedlichen Datenstrukturen, die Optionen zum Importieren von Daten und die Möglichkeiten der Anfragesprachen sowie der graphischen Oberflächen beider Systeme miteinander verglichen. Weiterhin wird in diesem Teil der Arbeit noch die interne Anfragenverarbeitung, die gebotenen Formen der Rückgabe von Ergebnissen und das Nutzer- und Projektmanagement der Systeme einander gegenüber gestellt. Ein weiterer wichtiger Aspekt des Vergleichs ist die Effizienz der Systeme bei unterschiedlichen Arten von Anfragen und unterschiedlich großen Datenmengen. Hierzu werden keine echten, sondern generierte medizinische Daten verwendet, die allerdings zu echten Daten ähnliche Charakteristika aufweisen.

1.1.2 Integration der Systeme

Im Rahmen dieser Arbeit soll außerdem eine Möglichkeit zur Integration beider Systeme geschaffen werden. Hierzu soll es möglich werden, Daten in das Format des jeweils anderen Systems umwandeln zu können. Des weiteren soll sowohl die Oberfläche von i2b2 genutzt werden können, um direkt auf den PaDaWaN-Datenbestand zuzugreifen, als auch eine Möglichkeit entstehen, mithilfe der PaDaWaN-Oberflächen und -Anfragesprache direkt Anfragen an Daten aus i2b2 zu senden. Zusammengefasst ist das Ziel dieser Arbeit, die wichtigsten Bestandteile beider Systeme miteinander kompatibel zu machen.

2 i2b2

i2b2 [3] ist eine Abkürzung für „Informatics for Integrating Biology and the Bedside“ und bietet Medizinern die Möglichkeit existierende medizinische Daten für Forschungszwecke auszuwerten. Es wird bereits international von vielen medizinischen Einrichtungen[1] eingesetzt. Im Folgenden wird zunächst ein kurzer Überblick über die Entstehung von i2b2 gegeben und anschließend dessen Aufbau und Funktionsweise genauer erläutert.

2.1 Entstehung

Im Jahr 2004 wurde das „i2b2 National Center for Biomedical Computing“ als Teil von „Partners HealthCare“[2] in Boston (USA) gegründet [4].

[1] Eine Liste mit vielen medizinischen Einrichtungen, an welchen i2b2 zum Einsatz kommt, kann unter https://www.i2b2.org/work/i2b2_installations.html eingesehen werden.

[2] http://www.partners.org

Die folgenden Erläuterungen zur Entwicklung von i2b2 basieren auf [5] und [6]. Die dort genannten Funktionen stellen nur einen Teil der gesamten Möglichkeiten von i2b2 dar und werden im weiteren Verlauf der Arbeit noch genauer erläutert werden.

Die allererste Version von i2b2 (1.0) wurde im November 2007 veröffentlicht. Sie umfasste bereits eine Möglichkeit, gemessene medizinische Werte mit bestimmten Metadaten (sogenannten Konzepten) wie zum Beispiel der Einheit der Messung in Verbindung zu bringen. Hierzu konnte ein ganzer Katalog an solchen Meta-Informationen angelegt werden. Die medizinischen Daten mussten in einer bestimmten Struktur in einer Datenbank gespeichert werden und konnten dann mithilfe der genannten Metadaten durchsucht werden. Alle obigen Daten konnten in unterschiedliche Projekte mit verschiedenen Nutzern, die wiederum unterschiedliche Zugriffsrechte hatten, organisiert werden. Zu diesem Zweck existierte eine web-basierte Administrationsoberfläche. Zusätzlich gab es zum Zugriff auf und zum Durchsuchen der Daten eine native Anwendung. Außerdem wurden Demo-Daten angeboten.
2008 gab es dann zusätzlich die Möglichkeit, mit einem Webclient auf die Daten zuzugreifen. Auch wurde eine Option zur Verwaltung kompletter Dateien in Verbindung mit den verschiedenen anderen Komponenten hinzugefügt. Des weiteren konnten Nutzer nun ihre am häufigsten genutzten Anfragen und Konzepte speichern und es wurde möglich nicht nur nach den genannten Metadaten zu suchen, sondern diese auch auf gewisse Werte einzuschränken.
Im Jahr 2010 erschienen die Versionen 1.4 und 1.5 und fügten vor allem verschiedene Plugins zur genaueren Auswertung sowie zum besseren Import der Daten hinzu. Auch wurde die Administrationsoberfläche überarbeitet und unter anderem um die Möglichkeit ergänzt, den Nutzern gewisse Rollen zuzuweisen und den Zugriff auf die Daten über diese Rollen zu verwalten. Die Daten konnten nun auch danach durchsucht werden, von welchem Arzt beziehungsweise von welcher Abteilung einer medizinischen Einrichtung sie erfasst wurden.
Mit Version 1.6, die von Oktober 2011 bis Ende 2013 aktuell war, wurden die Konzepte um sogenannte Modifier erweitert, mit denen zusätzliche Metadaten gespeichert werden konnten. Auch konnten nun komplette vorherige Suchanfragen als Teil einer neuen Anfrage genutzt werden.
Ende Dezember 2013 erschien dann Version 1.7, mit der es möglich wurde, Konzepte in einer bestimmten zeitlichen Abfolge zueinander zu suchen. Außerdem wurde eine Option zur Verwaltung von medizinischen Daten mit unter Datenschutzaspekten sensiblen Werten ergänzt. So können auch nicht anonymisierte Daten sicher gespeichert, anonymisiert ausgegeben und genau kontrolliert werden, wer auf diese Daten zugegriffen hat.
In den seit dem veröffentlichten Unterversionen von 1.7 wurden hauptsächlich Fehler behoben und nur noch kleinere Änderungen vorgenommen. Zum Beispiel können Nutzer mittlerweile ihr Passwort selber ändern und es ist möglich sich gewisse Suchergebnisse in graphischer Form anzeigen zu lassen.

Im Rahmen dieser Arbeit wird die Anfang 2016 erschienene Version 1.7.07 verwendet.

2.2 Zellenstruktur

i2b2 setzt sich aus mehreren sogenannten Zellen zusammen [7], die jeweils unterschiedliche Funktionalitäten abdecken und miteinander über festgelegte Schnittstellen kommunizieren. Ein Überblick über diese Struktur ist in Abbildung 1 zu sehen.

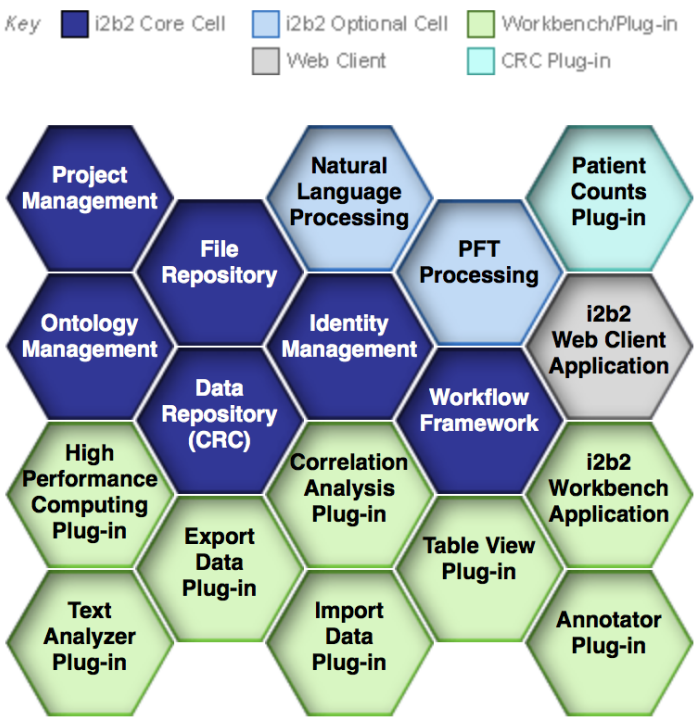

Abb. 1. Zellenstruktur von i2b2 (entnommen aus [7])

Im folgenden werden diese Zellen und ihre jeweiligen Funktionen genauer beschrieben.

2.2.1 Hive

Als „i2b2 Hive“ wird ganz allgemein der Aufbau und die Struktur von i2b2 bezeichnet. Darin arbeiten dann verschiedenste Komponenten zusammen, um unterschiedliche Arten des Zugriffs auf die dort hinterlegten medizinischen Daten zu ermöglichen. Die hierzu folgenden Erläuterungen basieren auf [8]. Im

Hive werden gewisse Funktionalitäten in separaten Einheiten (den Zellen) gruppiert und zeigen sich anderen Einheiten gegenüber in Form von einheitlichen Nachrichten, die zwischen den einzelnen Zellen ausgetauscht werden. Jede dieser Zellen kann jederzeit ausgetauscht werden, ohne das die Funktionalität der anderen beeinträchtigt wird. Das ist möglich, da die einzelnen Einheiten jeweils nur die Schnittstellen der anderen Zellen kennen und mit ihnen über Webdienste kommunizieren. Dadurch können die einzelnen Zellen auch problemlos auf unterschiedlichen Systemen an unterschiedlichen Standorten betrieben werden. Welche Schnittstellen aufgerufen werden, wird entweder manuell durch eine Nutzereingabe oder automatisch durch bestimmte Abläufe in einer Zelle gesteuert.
Eine einzelne Zelle kann als Dienst aus zwei Bestandteilen beschrieben werden. Zunächst gibt es die explizit definierte Transaktionskomponente, die nur eine sehr grundlegende Datenstruktur, aber dadurch eine eindeutige Kommunikation mit anderen Zellen, festlegt. Zusätzlich gibt es den Teil der Zelle, der von außerhalb nicht ersichtlich ist und die eigentliche Datenstruktur sowie den Zugriff auf diese Daten regelt. Da die Kommunikation untereinander nur über die Webschnittstelle stattfindet, kann dieser Teil der Zelle vollkommen beliebig und mit jeder möglichen Programmiersprache implementiert sein.

2.2.1.1 Kommunikation der i2b2-Zellen untereinander
Die oben erwähnten Schnittstellen müssen sogenannte RESTful-Web-Schnittstellen sein, die mittels XML-Nachrichten kommunizieren. In diesem Abschnitt wird die grundlegende Struktur dieser Nachrichten, die auch in Abbildung 2 zu sehen ist, basierend auf [9] genauer beschrieben. Sämtliche zwischen i2b2-Zellen ausgetauschte Nachrichten entsprechen diesem Format.

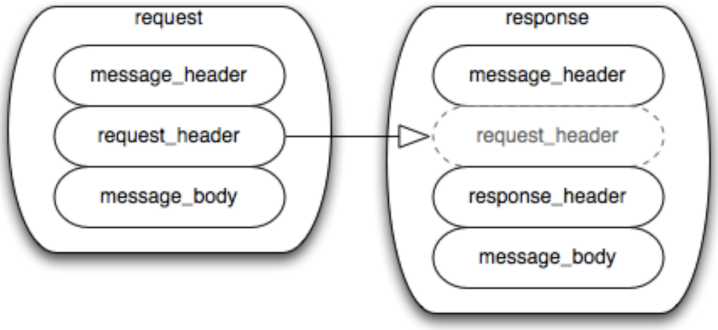

Abb. 2. Grundlegende Struktur der i2b2-XML-Nachrichten (entnommen aus [9])

Die i2b2-Nachrichten legen spezielle Eigenschaften fest, die die Zellen gemeinsam haben und die erforderlich sind, um die Nachrichten zu verschicken, zu empfangen und zu verarbeiten. Alle Anfragen werden mithilfe eines *<request>*-Tag verschickt und die Antwort wird mit einem *<response>*-Tag zurückgesendet.
Beide enthalten einen *<message_header>*-Tag, der bei Anfrage und Antwort

größtenteils identisch ist und sich aus diesen Bestandteilen zusammensetzt (basierend auf [10] und [11]):

- *<i2b2_version_compatible>* und *<hl7_version_compatible>* legen die Version des von dieser Nachricht verwendeten XML-Schemas fest.
- Mit *<sending_application>* und *<sending_facility>* sowie *<receiving_application>* und *<receiving_facility>* werden Name und Version der Softwarekomponente angegeben, die die Nachricht verschickt hat beziehungsweise an die sie gesendet wird.
- *<datetime_of_message>* gibt den Zeitpunkt an, zu dem die Nachricht verschickt wurde.
- Unter *<security>* wird der Login des aktuellen Nutzers mit dessen Nutzername und entweder direkt mit dessen Passwort oder mit einem Token angegeben, welches nur eine begrenzte Zeit lang gültig ist.
- Mittels *<message_type>* kann die Art sowie der Auslöser der übermittelten Nachricht noch genauer festgelegt werden.
- *<message_control_id>* enthält eine eindeutige Kennung der versendeten Nachricht, um zum Beispiel eine Antwort einer Anfrage zuordnen zu können.
- *<processing_id>* gibt an, wie genau die Nachricht verarbeitet werden soll. Zum Beispiel kann damit angegeben werden, ob die Nachricht zu Debugging-Zwecken oder im Produktivbetrieb geschickt wurde.
- *<accept_acknowledgement_type>* und *<application_acknowledgement_type>* legen fest, unter welchen Umständen eine Bestätigung der erhaltenen Nachricht erwartet wird. Das kann zum Beispiel immer oder nur bei einem Fehler der Fall sein.
- *<country_code>* enthält eine Länderkennung der Anwendung, die die Nachricht verschickt.
- Eine einzelne i2b2-Installation kann gleichzeitig mehrere Projekte mit unterschiedlichen Daten enthalten. Auf welches Projekt sich die aktuelle Nachricht bezieht, wird mit *<project_id>* angegeben.

In der Anfrage folgt auf den *<message_header>*, der *<request_header>*, der nur das Element *<result_waittime_ms>* enthält. Dieses gibt die Zeit an, in der die anfragende Anwendung eine Antwort erwartet.
Die Antwort kann optional den *<request_header>* der Anfrage erneut enthalten. Auf jeden Fall muss in ihr der *<response_header>*-Tag vorkommen, welcher einen der folgenden drei Statusmeldungen enthalten kann:

- **DONE** gibt an, das die Anfrage fertig verarbeitet wurde.
- **PENDING** wird zurückgegeben, wenn die Anfrage immer noch verarbeitet wird. In diesem Fall erhält man außerdem eine Zeitspanne zurück, nach welcher erneut überprüft werden kann, ob die Verarbeitung der Anfrage abgeschlossen wurde.
- Wenn **ERROR** zurückgeliefert wird, gab es einen Fehler bei der Anfragenverarbeitung.

Sowohl die Anfrage als auch die Antwort enthalten als letztes den *<message_body>*-Tag. Dieser ist vom allgemeinen i2b2-XML-Schema nicht weiter spezifiziert und kann durch die jeweiligen Zellen in beliebiger Form mit der eigentlichen Anfrage bzw. Antwort befüllt werden. In den folgenden Abschnitten werden dazu noch zwei Beispiele gezeigt. Der Inhalt dieses Tags unterscheidet sich sehr von Zelle zu Zelle und teilweise sogar innerhalb einer Zelle, sodass sich hier keine allgemeine Struktur angegeben lässt.

2.2.1.2 Ontology

Die „Ontology Management (abgekürzt ONT) Cell“ stellt eine der wichtigsten Komponenten von i2b2 dar. Sie verwaltet die komplette Terminologie, die in Verbindung mit fast allen anderen Zellen genutzt wird, sowie das komplette Wissen, das mit der Terminologie in Verbindung steht. Ein Beispiel für dieses Wissen wäre unter anderem der Datentyp, die Einheit oder auch Grenzwerte, ab denen ein Wert nicht mehr im Normalbereich liegt. Die folgenden Erläuterungen zu dieser Komponente basieren auf [12] und [13].

Datenbankstruktur

Bei i2b2 kann die komplette Terminologie der ONT-Zelle entweder in einer einzelnen Tabelle gespeichert, oder über mehrere Tabellen verteilt sein. Dadurch ist es möglich diese Metadaten zum Beispiel in Abhängigkeit von ihrem Datentyp in unterschiedlichen Tabellen abzulegen. Solche Datentypen wären zum Beispiel Diagnosen, und Laborergebnisse. Die Tabellen müssen aber immer dieselbe Struktur haben und aus den folgenden Spalten bestehen:

- **C_HLEVEL:** Die Terminologie kann in einer hierarchischen Struktur angelegt werden. Zum Beispiel kann sich auf oberster Ebene die Kategorie „Laborwerte“ befinden, die sich dann auf den unteren Ebenen in die Namen der unterschiedlichen konkreten Messungen aufteilt. In dieser Spalte wird die Nummer der Ebene eines Ausdrucks gespeichert, wobei die Nummerierung bei 0 beginnt.
- **C_FULLNAME:** Diese Spalte gibt den kompletten Pfad zu einem Ausdruck an. Die verschiedenen Hierarchieebenen sind in dieser Spalte dabei durch jeweils einen Backslash getrennt.
- **C_NAME:** Hiermit wird der Bezeichner der Terminologie gespeichert, der dann auch in der Benutzeroberfläche angezeigt wird.
- **C_SYNONYM_CD:** Diese Spalte enthält ein „Y“, wenn der Ausdruck ein Synonym für einen anderen ist, oder ein „N“, wenn es sich um den originalen Begriff handelt. Der Standardwert dieser Spalte ist „N“. Synonyme Ausdrücke haben den selben Basecode, der weiter unter erläutert wird.
- **C_VISUALATTRIBUTES:** Dies legt fest, wie ein Ausdruck in der Nutzeroberfläche dargestellt wird. Zu diesem Zweck setzt sich diese Spalte aus 3 Buchstaben zusammen:
  - Das erste Zeichen legt fest, ob der Term noch untergeordnete Ausdrücke hat. Wenn das der Fall ist, dann steht an dieser Stelle entweder ein „F“ (für „Folder“) oder ein „C“ (für „Container)“, je nachdem, ob dieser Ausdruck für eine Suche verwendet werden kann oder nicht. Wenn ein

Ausdruck keine untergeordneten Elemente mehr hat, so steht an dieser Stelle ein „L“ (für „Leaf“). Wenn der aktuelle Ausdruck ein sogenannter „Modifier“ für einen anderen ist, also zusätzliche Metainformationen zu einem Term enthält, so werden mit der selben Bedeutung wie oben die Buchstaben „D“, „O“ und „R“ benutzt. Zusätzlich gibt es noch „M“ für „Multiple“, was bedeutet, dass in der Terminologie mehrere Ausdrücke existieren, die zur Bedeutung des aktuellen Ausdrucks gehören.

- Mit dem zweiten Zeichen lässt sich festlegen, ob ein Ausdruck aktiv ist (mittels „A“), was bedeutet, dass er für Suchen verwendet werden kann. Alternativ kann ein Ausdruck auch inaktiv sein (mittels „I“), wodurch er nur ausgegraut angezeigt wird und nicht verwendet werden kann. Außerdem gibt es die Möglichkeit einen Ausdruck mittels „H“ an dieser Stelle komplett vor dem Nutzer zu verbergen.
- Das letzte Zeichen ist ein „E“, wenn ein Term bearbeitet werden kann. Das bedeutet auch, dass zu einem „Container“ oder „Folder“ untergeordnete Elemente ergänzt und der Term selbst gelöscht werden kann. Soll all das nicht möglich sein, so wird das dritte Zeichen weggelassen.

– **C_TOTALNUM:** Hiermit wird die Zahl an Patienten angegeben, die die aktuelle Terminologie bei einem der von ihnen gespeicherten Werte haben. Für „Modifier“ wird diese Spalte nicht genutzt, da sie sich auf mehr als einen Ausdruck beziehen können.
– **C_BASECODE:** Diese Spalte enthält eine eindeutige Kennung des aktuellen Konzepts. Diese kann in jedem beliebigen Format vorliegen.
– **C_METADATAXML:** Mit dieser Spalte werden zusätzliche Metadaten eines Ausdrucks in einer festgelegten XML-Form gespeichert. Das Folgende sind einige der möglichen Elemente dieser XML-Form:

  - *<DataType>* legt den Datentyp fest. Mögliche Werte sind „String“ für beliebigen Text wie zum Beispiel komplette Arztbriefe, „Enum“ für festgelegten Text (beispielsweise kann ein Eintrag für das Geschlecht eines Patienten nur die Werte „männlich“ oder „weiblich“ annehmen), oder „PosFloat“, „Float“, „PosInteger“ und „Integer“ für numerische Werte, wie zum Beispiel das Gewicht oder das Alter eines Patienten.
  - Mit *<MaxStringLength>* kann bei einem String-Datentyp die maximale Länge der Werte festgelegt werden.
  - *<LowofLowValue>*, *<HighofLowValue>*, *<LowofHighValue>*, *<HighofHighValue>*, *<LowofToxicValue>* und *<HighofToxicValue>* legen einen Bereich fest, in dem ein numerischer Wert als unbedenklich betrachtet werden kann. Durch diese vielen Parameter können auch „Übergangsbereiche“, in denen Werte anfangen bedenklich zu werden, sowie Grenzen, ab denen ein Wert als toxisch eingestuft wird, angegeben werden. Es kann auch nur eine beliebige Teilmenge dieser Grenzwerte genutzt werden.
  - Bei einem Enum-Datentyp, können unter *<EnumValues>* die möglichen Werte festgelegt werden, die dann bei einer Suche in der Nutzeroberfläche auch vorgeschlagen werden.

  - Unter *<UnitValues>* können beliebig viele mögliche Einheiten für einen Messwert angeben werden. Auch kann hiermit ein Umrechnungsfaktor zwischen zwei angegeben Einheiten gespeichert werden.

– **C_FACTTABLECOLUMN**, **C_TABLENAME**, **C_COLUMNNAME**, **C_OPERATOR** und **C_DIMCODE** werden alle genutzt, um das SQL-Select-Statement zusammen zu bauen, mit dem der Code eines Konzepts während einer Suche nach in i2b2 gespeicherten medizinischen Fakten abgefragt wird. Das Ergebnis dieser Anfrage muss somit immer ein Wert der i2b2-Fakten-Tabelle sein. Das Statement sieht dann wie folgt aus: „SELECT *C_FACTTABLECOLUMN* FROM *C_TABLENAME* WHERE *C_COLUMNNAME C_OPERATOR C_DIMCODE*“. Dabei ist *C_FACTTABLECOLUMN* die Spalte, deren Inhalt zurückgegeben werden soll, *C_TABLENAME*, der Name der zu durchsuchenden Konzept-Tabelle und *C_DIMCODE* der Wert, der verglichen mit *C_OPERATOR* in der Spalte *C_COLUMNNAME* gespeichert sein soll.
– **C_COLUMNDATATYPE:** Enthält entweder ein „T“, wenn dieser Eintrag zu Daten in Textform gehört, oder ein „N“ für numerische Daten.
– **C_COMMENT:** Diese Spalte kann optional beliebige Kommentare zu einem Ausdruck enthalten.
– **C_TOOLTIP:** Mit dieser Spalte wird der Tooltip gespeichert, der für jedes Konzept in der Nutzeroberfläche angezeigt wird. Normalerweise ist dies der Wert von *C_FULLNAME* mit Leerzeichen um jeden Backslash, damit er besser lesbar ist.
– **UPDATE_DATE**, **DOWNLOAD_DATE** und **IMPORT_DATE** geben an, wann das aktuelle Konzept zuletzt aktualisiert beziehungsweise heruntergeladen oder importiert wurde.
– **SOURCESYSTEM_CD:** Hiermit wird eine Kennung des Systems gespeichert, aus dem ein Ausdruck übernommen wurde.
– **VALUETYPE_CD:** In dieser Spalte steht ein Code für die Art des aktuellen Konzepts. Zurzeit kann dies entweder *DOC* (für Dokumente oder Notizen) oder *LAB* (für Laborwerte) sein. Dieser Wert kann auch weggelassen werden.
– **M_APPLIED_PATH:** Mit dieser Spalte kann bei Modifiern der Pfad des Konzepts (oder mithilfe eines %-Zeichens am Ende des Pfads auch mehrere Konzepte) angegeben werden, auf das sie sich beziehen. Bei Konzepten selbst enthält diese Spalte in der Regel den Wert „@“, was angibt, dass diese Spalte nicht genutzt wird.
– **M_EXCLUSION_CD:** Wenn in dieser Spalte anstelle von „null“ ein „X“ steht, so wird damit angegeben, dass sich ein Modifier explizit nicht auf das beziehungsweise die von *M_APPLIED_PATH* angegebene Konzept(e) bezieht.
– **C_PATH:** Diese Spalte ist ein Teil von *C_FULLNAME*. Sie enthält den Wert von *C_FULLNAME* des Eltern-Elements des aktuellen Konzepts in der Hierarchie und ergibt zusammen mit *C_SYMBOL* (siehe unten) wieder den *C_FULLNAME* des aktuellen Konzepts.

– **C_SYMBOL:** In dieser Spalte steht eine eindeutige Abkürzung des *C_NAME*, die Zusammen mit *C_PATH* den *C_FULLNAME* ergibt.

Ein Beispiel für einen konkreten Eintrag in dessen XML-Form ist in Listing 1 zu sehen.

```
<modifier>
  <level>1</level>
  <applied_path>\i2b2\Medications\%</applied_path>
  <key>\\i2b2_MEDS\Medication\Route\</key>
  <fullname>\Medication\Route\</fullname>
  <name>Route</name>
  <visualattributes>RA </visualattributes>
  <synonym_cd>N</synonym_cd>
  <totalnum>21</totalnum>
  <basecode>MED:ROUTE</basecode>
  <metadataxml>
    <ValueMetadata>
      <DataType>Enum</DataType>
      <EnumValues>
        <Val>inhalation</Val>
        <Val>injection</Val>
        <Val description="Intravenous">IV</Val>
        <Val description="By Mouth">PO</Val>
        <Val description="By Rectum">PR</Val>
        <Val>topical</Val>
        <Val>transdermal</Val>
      </EnumValues>
    </ValueMetadata>
  </metadataxml>
  <facttablecolumn>MODIFIER_CD</facttablecolumn>
  <tablename>MODIFIER_DIMENSION</tablename>
  <columnname>MODIFIER_PATH</columnname>
  <columndatatype>T</columndatatype>
  <operator>LIKE</operator>
  <dimcode>\Medication\Route\</dimcode>
  <tooltip>Medication Route</tooltip>
</modifier>
```

Listing 1. Beispiel für einen Konzept-Modifier in dessen XML-Form

Dieser Eintrag ist ein Modifier, der verwendet wird, um zu einzelnen Medikamenten angeben zu können, wie sie verabreicht werden. Da dies nur auf ganz bestimmten Wegen möglich ist, wird der Datentyp *Enum* verwendet. Zu sehen ist außerdem, dass sich der Eintrag auf der zweiten Hierarchieebene direkt unter dem Eintrag für Medikamente selbst befindet und auf alle Einträge unter diesem Obereintrag „Medication“ angewendet werden kann. Der Modifier kommt im aktuellen Datenbestand 21 mal vor und ist kein Synonym für einen anderen

Eintrag. Anhand des Wertes von *visualattributes* ist außerdem ersichtlich, dass der Modifier keine untergeordneten Elemente mehr enthält, in Anfragen verwendet und nicht über die graphischen Benutzeroberflächen bearbeitet werden kann.

Mögliche Anfragen

Die Ontology Management Komponente von i2b2 unterstützt verschiedene Anfragen. Die Wichtigsten werden im Folgenden vorgestellt:

- Mit **get_categories** werden die Konzepte auf der höchsten Hierarchieebene abgefragt. Wenn diese auf mehrere Tabellen verteilt sind, liefert diese Anfrage in der Regel die Bezeichner der einzelnen Tabellen zurück. Auch wird der angemeldete Nutzer berücksichtigt, sodass nur Konzepte angezeigt werden, auf die dieser auch zugreifen darf. Die Anfrage erfordert keine speziellen Parameter. Es kann aber - wie auch bei den meisten der folgenden Anfragen - festgelegt werden, ob alle der oben beschriebenen Felder eines Konzepts zurückgegeben werden sollen, oder nur eine Teilmenge davon. Diese Anfrage wird nach Starten der i2b2-Benutzeroberfläche in der Regel als Erstes gestellt, um die Ansicht, aus denen die Konzepte in die Suche gezogen werden können, mit den Konzepten auf höchster Ebene zu befüllen.
- Die Anfrage **get_children** wird verwendet, um alle Konzepte zu erhalten, die in der Hierarchie direkt an einem vorgegebenen Eltern-Konzept hängen. Hierzu wird der Pfad des Eltern-Elements als Parameter benötigt. Bei dieser, sowie bei vielen der folgenden Anfragen kann außerdem festgelegt werden, ob auch Konzepte, die als Synonym eines anderen markiert sind, zurückgegeben werden sollen. Auch kann festgelegt werden, ob Konzepte, die als versteckt markiert sind, in der Rückgabe enthalten sein sollen.
- **get_name_info** ermöglicht ein Durchsuchen der *C_NAME*-Spalte. Dazu wird als Parameter der Suchbegriff sowie eine Suchstrategie vorgegeben. Diese Strategie kann eine der folgenden sein:
  - **contains** legt fest, dass der Suchbegriff nur an beliebiger Stelle im Namen enthalten sein muss.
  - **exact** wird verwendet, wenn Suchbegriff und Name exakt identisch sein sollen.
  - **left** bedeutet, dass der Name mit dem Suchbegriff beginnen muss.
  - **right** meint, dass der Name mit dem Suchbegriff enden soll.

  Die Anfrage liefert dann alle passenden Suchergebnisse zurück.
- **get_term_info** kann verwendet werden, um sich zusätzliche Felder zu einem bestimmten Konzept zurückgeben zu lassen. Hierzu wird als Parameter der Pfad des Konzepts benötigt.
- **get_schemes** wird verwendet, um alle zum Kodieren der Konzepte verwendeten Schemata abzufragen. Dies wird in der Regel für die nächste Anfrage benutzt. So kann der Nutzer in der Oberfläche direkt ein Kodierungsschema auswählen und dann innerhalb von diesem nach bestimmten Codes suchen.
- Die Anfrage **get_code_info** wird benutzt, um die *C_BASECODE*-Spalte zu durchsuchen. Ein solcher Code hat in der Regel die Form „*Kodierungsschema*:*Code*“und wird der Anfrage als Parameter übergeben.

- **add_child** fügt ein neues Konzept zur Hierarchie hinzu. Dies ist nur ab der zweiten Hierarchieebene möglich, da die Oberste in der Regel die verschiedenen Datenbanktabellen der Konzepte darstellt.
- **delete_child** löscht ein Konzept aus der Hierarchie. Bei dieser Anfrage kann außerdem angegeben werden, ob alle untergeordneten Konzepte mit gelöscht werden sollen.
- Mit **modify_child** ist es möglich, die Werte eines Konzepts zu verändern.
- Die Anfrage **update_crc_concept** wird benutzt, um der CRC-Komponente mitzuteilen, dass sie ihre eigenen Konzepttabelle anhand der Daten der ONT-Komponente aktualisieren soll. Hierbei kann angegeben werden, ob die CRC-Konzepttabelle komplett neu aufgebaut werden soll oder nur seit der letzten Synchronisation hinzugekommene Konzepte berücksichtigt werden sollen. Dies ist nötig, da in der genannten CRC-Tabelle ein Teil der ONT-Daten ebenfalls gespeichert sein müssen, um so während einer Suchanfrage dem Pfad eines zu suchenden Konzepts dessen Basecode zuordnen zu können, ohne die ONT-Zelle zu kontaktieren.
- **update_concept_totalnum** startet eine Synchronisierung mit der CRC-Komponente, bei der die Werte der *C_TOTALNUM*-Spalte aktualisiert werden. Hierbei können entweder alle Werte von *C_TOTALNUM* aktualisiert werden, oder nur die, die noch keinen Wert (also den Wert *NULL*) haben.
- **get_ont_process_status** ermöglicht es, den aktuellen Stand eines Synchronisationsprozesses abzufragen, der mit einer der letzten beiden Anfragen gestartet wurde.
- Mit **get_dirty_state** ist es möglich abzufragen, ob ein und wenn ja welcher Aufruf von *update_crc_concept* nötig ist. Dazu liefert diese Anfrage zurück, ob seit der letzten Synchronisation nur neue Konzepte hinzugekommen sind, oder ob auch bestehende Konzepte bearbeitet und/oder gelöscht wurden, wodurch eine komplette Neuerstellung der CRC-Konzepttabelle nötig wird.
- **get_modifiers** erlaubt die Abfrage aller für ein Konzept existierender Modifier anhand des vollständigen Pfads des Konzepts.
- Durch **get_modifer_info** können zu einem konkreten und durch dessen Pfad angegebenen Modifier zusätzliche Felder abgefragt werden.
- **get_modifier_children** fragt alle Kinder-Modifier zu einem durch dessen Pfad angegebenen Eltern-Modifier ab.
- **get_modifier_name_info** erlaubt das Durchsuchen der *C_NAME*-Spalte für Modifier mit den selben Parametern wie *get_name_info*.
- **get_modifier_code_info** durchsucht analog zu *get_code_info* die *C_BASECODE*-Spalte für Modifier.
- **add_modifier** ergänzt die Konzept-Hierarchie um einen zusätzlichen Modifier.

Alle diese Anfragen werden als Teil des *<message_body>*-Tags angegeben, der wiederum selbst vom oben beschriebenen i2b2-XML-Schema festgelegt wird. Eine *get_categories*-Anfrage sähe dann zum Beispiel so aus, wie es in Listing 2 zu sehen ist.

```
<get_categories synonyms="true" hiddens="false" type="core"/>
```

Listing 2. Beispielhafte *get_ categories*-Anfrage

Darin wird festgelegt, dass zwar Synonyme, aber keine versteckten Einträge zurückgegeben werden sollen. Der *type* gibt an, wie viele der oben beschriebenen Spalten zurückgegeben werden sollen. „core“ an dieser Stelle sorgt dafür, dass sämtliche bis auf nur intern genutzte Spalten (wie zum Beispiel *SOURCESYSTEM_ CD* oder *IMPORT_ DATE*) ausgegeben werden.

2.2.1.3 CRC

Eine weitere zentrale Komponente von i2b2 ist die „Data Repository Cell“. Als alternativer Ausdruck zu „data repository“ wird auch „Clinical Research Chart“ (abgekürzt CRC) benutzt. Mit dieser Zelle werden die eigentlichen Patientendaten gespeichert, verwaltet und durchsucht. Normalerweise wird mit dieser Komponente zunächst ein Satz an Patienten, der zu gewissen Konzepten passt, gesucht. Anschließend werden für diesen Satz an Patienten die konkreten Messungen zu den Konzepten abgefragt. Dieser Ablauf wird weiter unten noch genauer anhand eines Beispiels beschrieben. Ein weiterer Einsatzzweck dieser Zelle ist das Importieren neuer Fakten in i2b2. Im folgenden wird diese Komponente basierend auf [14] und [15] genauer erläutert.

Datenbankstruktur

Die CRC-Komponente setzt sich aus mehreren Tabellen zusammen. Dabei gibt es Tabellen, in denen konkrete Patientendaten gespeichert werden (Patient, Visit und Observation), Tabellen, in denen weitere Informationen zu in den Datentabellen verwendeten Codes nachgeschlagen werden können (Concept, Provider und Code), sowie Tabellen, die dazu dienen, Kennungen aus anderen Systemen i2b2-Kennungen zuzuweisen (Patient mapping und Visit mapping). Neben dieser Aufteilung der Tabellen nach Art der enthaltenen Daten lassen sich die Tabellen auch noch nach ihren Beziehungen untereinander beschreiben. Diese Struktur folgt einem Sternschema, da es die zentrale Faktentabelle gibt, die mit den anderen Tabellen verbunden ist, welche zusätzliche Informationen zu bei den Fakten verwendeten Codes und Kennungen enthalten. Diese anderen Tabellen werden auch als Dimensionstabellen bezeichnet.

Alle Tabellen der CRC-Zelle haben die folgenden Spalten:

- **UPDATE_DATE** ist das Datum sowie die Uhrzeit, an dem eine Spalte zuletzt vom System, aus dem die Daten stammen, aktualisiert wurde.
- In **DOWNLOAD_DATE** wird der Zeitpunkt gespeichert, an dem die Daten vom Quellsystem heruntergeladen wurden.
- Am **IMPORT_DATE** wurde der Datensatz erstmals in i2b2 importiert.
- **SOURCESYSTEM_CD** speichert eine eindeutige Kennung des Systems, aus dem die Daten importiert wurden.
- Mit **UPLOAD_ID** kann eine numerische Kennung des Uploadvorgangs in i2b2 gespeichert werden.

Nun wird der Aufbau aller Tabellen genauer beschrieben:

- **OBSERVATION_FACT:** Dies ist die zentrale Faktentabelle des Sternschemas. Alle der folgenden Spalten müssen in jeder Implementierung einer CRC-Zelle vorhanden sein. Die ersten 5 Spalten bilden zusammen den Primary Key dieser Tabelle.
  - **ENCOUNTER_NUM** enthält eine Kennung des Besuch eines Patienten, bei dem der Fakt angelegt wurde.
  - Die eindeutige Kennung des Patienten wird in **PATIENT_NUM** gespeichert.
  - **CONCEPT_CD** speichert die eindeutige Kennung des bei einem Fakt verwendeten Konzepts. Diese Kennung entspricht in der Regel dem *C_BASECODE* der ONT-Zelle.
  - In **PROVIDER_ID** steht eine Kennung des Arztes oder der Abteilung einer medizinischen Einrichtung, die den aktuellen Fakt angelegt hat.
  - **START_DATE** ist der Zeitpunkt, zu dem der Fakt beobachtet und/oder im System angelegt wurde.
  - **MODIFIER_CD** enhtält optional eine Kennung eines verwendeten Modifier. Wenn zum Beispiel der Modifier „Dosierung“ benutzt wird, so wäre der Wert des Faktes die verwendete Dosierung.
  - Die **INSTANCE_NUM** erlaubt es für eine *CONCEPT_CD* mehrere Modifier zu verwenden. In diesem Fall unterscheidet sich dann nur die *MODIFIER_CD* und die *INSTANCE_NUM* bleibt je nach verwendeter Kodierung gleich oder ähnlich. Soll diese Möglichkeit genutzt werden, so wird in der Regel der Primary Key um diese sowie um die *MODIFIER_CD*-Spalte erweitert.
  - **VALTYPE_CD** legt fest, welche Art von Wert bei diesem Fakt gespeichert wird. Hierfür kann sie einen der folgenden Werte annehmen:
    - **@ oder NULL**, wenn kein Wert gespeichert wird. Das kann zum Beispiel benutzt werden um nur anzugeben, dass ein Konzept vorhanden ist.
    - **N** bei numerischen Werten wie zum Beispiel Laborergebnissen.
    - **T** bei kurzen Texten als Wert. Ein Beispiel hierfür wären Konzepte, die nur einen von gewissen vorgegebenen Werten annehmen können.
    - **B** bei beliebigen Texten wie zum Beispiel Notizen oder einem Arztbrief.
    - **NLP**, wenn das XML-Ergebnis einer durchgeführten Sprachverarbeitung als Wert gespeichert werden soll.
  - **TVAL_CHAR** speichert bei einem *T* in der vorherigen Spalte den Wert des Fakts sowie bei einem *N* den Operator, der mit dem numerischen Wert in Verbindung steht. Das kann unter anderem *E* sein, wenn der gemessene Wert genau dem gespeicherten Wert entspricht, oder *L*, wenn mit dem Fakt nur angegeben werden soll, dass der beobachtete Wert kleiner als der gespeicherte ist.
  - **NVAL_NUM** speichert bei einem *N* als *VALTYPE_CD* den konkreten numerischen Wert.

- Mit einem *X* in der Spalte **VALUEFLAG_CD** kann angegeben werden, dass bei *VALTYPE_CD B* oder *NLP* der Wert in verschlüsselter Form gespeichert ist. Bei einem *VALTYPE_CD* von *N* oder *T* kann in dieser Spalte ein *H*, *L* oder *A* stehen, um anzugeben, dass der Wert hoch, niedrig oder einfach nur in irgendeiner Form auffällig ist.
- Die Spalte **QUANTITY_NUM** gibt die Menge des Wertes in der *NVAL_NUM*-Spalte an.
- **UNITS_CD** speichert die Einheit des Wertes in der *NVAL_NUM*-Spalte. Dies wird in der Regel in Verbindung mit Laborwerten genutzt.
- Mit **END_DATE** ist es möglich, den Endzeitpunkt eines Fakts zu speichern, falls es sich bei diesem nicht um eine einzelne Messung, sondern um einen Zeitraum handelt.
- **LOCATION_CD** erlaubt das Speichern einer Kennung des Ortes, an dem der Fakt gemessen wurde. Dies kann zum Beispiel der Name eines Krankenhauses sein.
- Durch die Spalte **CONFIDENCE_NUM** ist es möglich eine Beurteilung der Genauigkeit eines gemessenen Wertes abzuspeichern.
- In der Spalte **OBSERVATION_BLOB** wird bei einem *VALTYPE_CD* von *B* oder *NLP* der eigentliche Wert abgespeichert. Bei den anderen möglichen Werten von *VALTYPE_CD* können hier beliebige zusätzliche Daten zu einem Fakt gespeichert werden.

– **PATIENT_DIMENSION:** Jede Zeile dieser Tabelle stellt einen Patienten im System dar. Nur die ersten vier der folgenden Spalten sind für die CRC-Komponente zwingend erforderlich. Alle anderen stellen nur eine Beispielkonfiguration dar. Sie können beliebig geändert, gelöscht und ergänzt werden. Dadurch ist es möglich unterschiedlichste demographische Daten für einen Patienten zu speichern.
  - Die **PATIENT_NUM**-Spalte stellt den Primary Key dieser Tabelle dar und enthält eine eindeutige Kennung des Patienten.
  - In **BIRTH_DATE** steht - falls im System vorhanden - das Geburtsdatum des Patienten.
  - **DEATH_DATE** enthält - falls vorhanden - den Todeszeitpunkt.
  - **VITAL_STATUS_CD** ist ein Code aus zwei Buchstaben, der die Genauigkeit des Geburts- und Todeszeitpunkts angibt. Dabei gibt der erste Buchstabe an, ob überhaupt bekannt ist, dass ein Patient gestorben ist. Dies ist auch möglich, wenn kein Todeszeitpunkt bekannt ist. Falls ein solcher bekannt ist, dann gibt dieses erste Zeichen an, wie genau der Wert in *DEATH_DATE* ist. Hier ist eine beliebige Abstufung von einer Sekundengenauigkeit bis hin dazu, dass nur das Todesjahr bekannt ist, möglich. Das zweite Zeichen gibt analog die Genauigkeit des Wertes unter *BIRTH_DATE* an.
  - Die folgenden Spalten sind die oben erwähnten beispielhafte Möglichkeiten zum Speichern weiterer demographischer Daten eines Patienten: **SEX_CD**, **AGE_IN_YEARS_NUM**, **LANGUAGE_CD**, **MARITAL_STATUS_CD**, **RELIGION_CD** und **ZIP_CD**. Mit

ihnen wäre es zum Beispiel möglich das Geschlecht, das Alter, die gesprochene Sprache, den Familienstand, die Religion sowie die Postleitzahl eines Patienten zu speichern.
    - Die Spalte **PATIENT_BLOB** ist ebenfalls optional und wird in der Regel verwendet, um zusätzliche Daten eines Patienten in beliebiger Form zu speichern.
- **VISIT_DIMENSION:** Diese Tabelle ist ähnlich zur *PATIENT_DIMENSION*-Tabelle aufgebaut. Sie enthält pro Zeile einen Besuch eines Patienten. Das kann zum Beispiel ein Krankenhausaufenthalt oder der Besuch in einer Praxis sein. Auch in dieser Tabelle sind nur die ersten fünf Spalten zwingend erforderlich. Die restlichen können wieder beliebig angepasst werden, um zusätzliche Daten eines Besuchs zu speichern. Die ersten beiden Spalten bilden zusammen den Primary Key.
    - **ENCOUNTER_NUM** enthält eine eindeutige Kennung des Besuchs.
    - **PATIENT_NUM** speichert die eindeutige Kennung des zugehörigen Patienten.
    - **START_DATE** speichert den Anfangszeitpunkt des Besuchs.
    - **END_DATE** enthält den Abschlusszeitpunkt eines Besuchs.
    - Die Spalte **ACTIVE_STATUS_CD** funktioniert analog zur *VITAL_STATUS_CD*-Spalte der *PATIENT_DIMENSION*-Tabelle und enthält einen Code aus zwei Buchstaben, der die Genauigkeit von *END_DATE* (erstes Zeichen) und *START_DATE* (zweites Zeichen) angibt.
    - Die Spalten **INOUT_CD** und **LOCATION_CD** sind Beispiele für die erwähnten optionalen Spalten. Hiermit könnte zum Beispiel gespeichert werden, ob ein Besuch noch andauert (oder der Patient schon wieder entlassen wurde) und in welcher medizinischen Einrichtung der Besuch stattfand.
    - Die optionale Spalte **VISIT_BLOB** bietet die Möglichkeit, beliebige zusätzliche Daten eines Besuchs zu speichern.
- **CONCEPT_DIMENSION:** In dieser Tabelle wird pro Zeile ein Konzept gespeichert. Sie setzt sich aus den folgenden Spalten zusammen:
    - **CONCEPT_PATH** enthält den vollständigen Pfad eines Konzepts in der Konzepthierarchie und entspricht in der Regel dem Wert von *C_FULLNAME* in der Ontology-Tabelle. Diese Spalte stellt außerdem den Primary Key der *CONCEPT_DIMENSION*-Tabelle dar.
    - In **CONCEPT_CD** wird eine eindeutige Kennung des Konzepts gespeichert, welche in der Regel dem *C_BASECODE* der ONT-Zelle entspricht.
    - **NAME_CHAR** speichert den Namen des Konzepts und entspricht somit in der Regel der ONT-Spalte *C_NAME*.
    - Die optionale Spalte *CONCEPT_BLOB* erlaubt das Speichern beliebiger zusätzlicher Daten zu einem Konzept.
- **PROVIDER_DIMENSION:** Mithilfe dieser Tabelle werden die sogenannten Provider gespeichert. Damit sind die Ärzte und medizinischen Abteilungen einer Einrichtung gemeint, von denen die einzelnen Patientendaten erfasst wurden. Die ersten beiden der folgenden Spalten bilden zusammen den Primary Key dieser Tabelle.

  - In **PROVIDER_ID** wird eine eindeutige Kennung des Providers gespeichert.
  - **PROVIDER_PATH** enthält den kompletten „Pfad“ zu einem Provider. Dadurch ist es möglich zu sehen, wie dieser sich in die Organisationshierarchie einer medizinischen Einrichtung einordnen lässt. Dieser Pfad kann sich zum Beispiel aus dem Namen des Krankenhauses, dem der Abteilung sowie dem des behandelnden Arztes zusammen setzen.
  - **NAME_CHAR** speichert den Namen des aktuellen Providers.
  - Mit der optionalen Spalte **PROVIDER_BLOB** können beliebige zusätzliche Daten eines Providers gespeichert werden.
- **MODIFIER_DIMENSION:** Diese Tabelle hat eine sehr ähnliche Struktur zur *CONCEPT_DIMENSION*-Tabelle. In ihr wird pro Zeile ein Modifier gespeichert. Die Zugehörigkeit der Modifier zu den Konzepten wird über die ONT-Komponente verwaltet.
  - In **MODIFIER_PATH** wird der komplette Pfad des Modifiers in der Konzepthierarchie gespeichert. Dieser Wert entspricht in der Regel dem *C_FULLNAME* aus der Ontology-Tabelle. Diese Spalte ist außerdem der Primary Key der *MODIFIER_DIMENSION*-Tabelle.
  - **MODIFIER_CD** enthält eine eindeutige Kennung des Modifiers. Diese ist in der Regel identisch zum *C_BASECODE* der ONT-Zelle.
  - **NAME_CHAR** speichert den Namen des Modifiers, welcher in der Regel der ONT-Spalte *C_NAME* entspricht.
  - Die optionale Spalte *MODIFIER_BLOB* erlaubt auch hier das Speichern beliebiger zusätzlicher Daten zu einem Modifier.
- **CODE_LOOKUP:** Diese Tabelle speichert den Wert hinter den verschiedenen Codes, die in den anderen Tabellen benutzt wurden. Die ersten der drei folgenden Spalten bilden zusammen den Primary Key dieser Tabelle.
  - **TABLE_CD** enthält den Namen der Tabelle, in welcher der Code verwendet wird.
  - Mit **COLUMN_CD** wird der Name der Spalte gespeichert, in der der Code vorkommt.
  - **CODE_CD** enthält dann den eigentlichen Code.
  - **NAME_CHAR** speichert die Bedeutung des Codes.
  - **Beispieldatensatz:** In der *VISIT_DIMENSION*-Tabelle (Wert von *TABLE_CD*) kann es die optionale Spalte *LOCATION_CD* (Wert von *COLUMN_CD*) geben, um eine Kennung der medizinischen Einrichtung zu speichern, in der der Besuch stattfand. Diese könnte dann zum Beispiel den Wert „MGH“ (Wert von *CODE_CD*) enthalten. Um die medizinische Einrichtung hinter diesem Code herausfinden zu können, würde für diesen Datensatz dann in der *NAME_CHAR*-Spalte der *CODE_LOOKUP*-Tabelle der Wert „Massachusetts General Hospital“ stehen.
  - Seit i2b2-Version 1.6.00 werden in dieser Tabelle auch die **Beschreibungen** für optionale Spalten anderer Tabellen gespeichert. In diesem Fall hat die *CODE_CD*-Spalte den Wert *CRC_COLUMN_DESCRIPTOR* und *NAME_CHAR* enthält die eigentliche Beschreibung für die durch

die anderen beiden Spalten der *CODE_LOOKUP*-Tabelle angegebene optionale Spalte.
    - Mit der optionalen Spalte **LOOKUP_BLOB** können beliebige zusätzliche Daten zu einem Eintrag gespeichert werden.
- **PATIENT_MAPPING:** Mithilfe dieser Tabelle können Patientenkennungen aus anderen System zusätzlich zur i2b2-eigenen Patientenkennung gespeichert werden. Die ersten beiden der folgenden Spalten bilden zusammen den Primary Key.
    - **PATIENT_IDE** ist die aus dem ursprünglichen System übernommene Patientenkennung. Diese kann beliebige Zeichen enthalten.
    - **PATIENT_IDE_SOURCE** enthält den Namen des ursprünglichen Systems.
    - **PATIENT_NUM** speichert die zugehörige i2b2-Patientenkennung, die immer rein numerisch ist.
    - Mit **PATIENT_IDE_STATUS** ist es möglich den Status dieser Patientenkennung im Quellsystem zu speichern. Mögliche Werte wären zum Beispiel „aktiv“, „inaktiv“ oder „gelöscht“.
    - Die Spalte **PROJECT_ID** erlaubt das Speichern einer zusätzlichen Projektkennung zu der obigen Zuweisung.
- **ENCOUNTER_MAPPING:** Diese Tabelle erlaubt es Kennungen für Besuche aus anderen Systemen ergänzend zu den i2b2-eigenen Besuchskennungen zu speichern. Die ersten zwei der folgenden Spalten bilden zusammen den Primary Key. Sie ist sehr ähnlich zur *PATIENT_MAPPING*-Tabelle aufgebaut.
    - In **ENCOUNTER_IDE** steht die aus dem ursprünglichen System übernommene Besuchskennung. Sie kann beliebige Zeichen enthalten.
    - **ENCOUNTER_IDE_SOURCE** ist der Name des ursprünglichen Systems.
    - **ENCOUNTER_NUM** enthält die zugehörige i2b2-eigene Kennung des Besuchs, die immer nur aus Zahlen besteht..
    - Durch **ENCOUNTER_IDE_STATUS** kann der Status einer Besuchskennung in deren Quellsystem zu speichern. Mögliche Werte sind unter anderem „aktiv“, „inaktiv“ oder „gelöscht“.
    - Die Spalte **PROJECT_ID** ermöglicht das Speichern einer zusätzlichen Projektkennung zu dieser Zuweisung der beiden Besuchskennungen.

Wenn unter den oben erläuterten Spalten zwei in unterschiedlichen Tabellen denselben Namen haben, so ist es in i2b2 immer möglich einen SQL-Join dieser Tabellen durchzuführen, um so zusätzliche Informationen zu einem Eintrag zu erhalten. So kann zum Beispiel die *ENCOUNTER_NUM*-Spalte der *OBSERVATION_FACT*-Tabelle mit der gleichnamigen Spalte der *VISIT_DIMENSION*-Tabelle verbunden werden, um zusätzliche Informationen zu dem Besuch, bei dem ein Fakt angelegt wurde, abzufragen.
Aus den Werten der obigen Tabellen lassen sich in i2b2 sogenannte „PATIENT DATA OBJECTs“ (PDOs) zusammenbauen. Dies ist in der Regel eine XML-Darstellung der obigen Struktur und kann jede beliebige Teilmenge der gespeicherten Daten enthalten. Ein PDO kann auch verwendet werden, um neue Daten

in i2b2 zu speichern. Hierzu müssen die Daten eines anderen Systems in PDO-Form exportiert werden. Die so exportierte Datei lässt sich anschließend mithilfe der CRC-Komponente in i2b2 wieder importieren, wobei die i2b2-Tabellen entsprechend befüllt werden. Die Importmöglichkeiten in i2b2 werden im weiteren Verlauf dieser Arbeit noch genauer beschrieben.

Mögliche Anfragen

Ganz allgemein gibt es zwei Arten von Nutzern der i2b2-CRC-Komponente. Einerseits ist das der sogenannte „Contributing"-Nutzer, also jemand, der neue Patientendaten zur CRC-Komponente hochlädt oder ältere Daten löscht. Andererseits gibt es den „Browsing"-Nutzer. Dieser hat vier mögliche Interaktionen mit der CRC-Zelle: Er kann Anfragen definieren, um eine Teilmenge der Patienten zu erhalten, er kann frühere Anfragen durchsehen, solche Anfragen erneut ausführen und außerdem konkrete Daten von Patienten abfragen. Das Erstellen einer Anfrage erfordert immer zunächst eine Interaktion mit der ONT-Komponente.
In einem möglichen Einsatzszenario könnte ein Nutzer das Alter sowie das Geschlecht aller Patienten abfragen wollen, die entweder Diabetes oder Asthma haben. Dazu sind die folgenden Schritte notwendig, die auch in Abbildung 3 zu sehen sind.

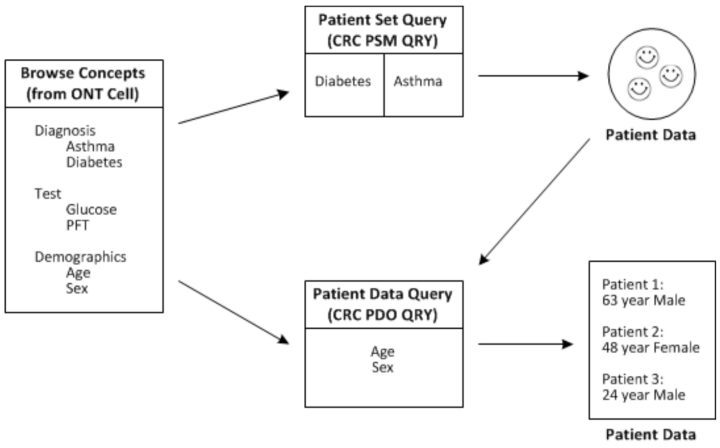

Abb. 3. Mögliches Einsatzszenario der CRC-Komponente (entnommen aus [14])

- Zunächst werden die gesuchten Diagnosen (Diabetes und Asthma) aus der ONT-Zelle ausgewählt. Mit diesen wird der sogenannte „Patient Set Query" definiert, der die zu dieser Anfrage gehörende Teilmenge an Patienten in der CRC-Komponente speichert.
- Anschließend werden die gewünschten demographischen Konzepte (Alter und Geschlecht) ausgewählt, die dann zusammen mit obigem „Patient Set" den sogenannten „Patient Data Query" bilden.

– Diese letzte Anfrage liefert dann Alter und Geschlecht aller Patienten mit Diabetes oder Asthma zurück.

Ein Patient Set Query wird in der CRC-Komponente in einer hierarchischen Struktur in der Regel auch in einer Datenbank gespeichert. Diese Struktur setzt sich aus den folgenden Bestandteilen zusammen:

– Als **QueryMaster** wird das Element bezeichnet, in welchem die eigentliche Anfrage gespeichert wird. Daneben enthält es auch noch den Nutzer, der die Anfrage erstellt hat sowie den Zeitpunkt dieser Erstellung und die Gruppe des Nutzers. Auch der Name der Anfrage wird hiermit gespeichert.
– Ein QueryMaster hat dann auf der nächsten Ebene beliebig viele **QueryInstance**-Objekte, die jeweils einer Ausführung der Anfrage entsprechen und deren Dauer und Status (zum Beispiel, ob die Anfrage noch läuft oder schon abgeschlossen ist) speichern.
– Auf unterster Ebene enthält dann jede QueryInstance beliebig viele **QueryResult**-Elemente. In diesen wird das Ergebnis der Anfrage gespeichert. Das ist dann in der Regel eine Kennung des gefundenen „Patient Set“ sowie die Zahl der darin enthaltenen Patienten. Wenn etwas anderes abgefragt wurde (die Möglichkeiten dazu werden weiter unten noch beschrieben), so kann ein QueryResult auch andere Daten enthalten.

Die Anfragen selbst setzen sich aus sogenannten **Panels** zusammen, was auch in Abbildung 4 zu sehen ist und im Folgenden genauer erläutert wird. Die verschiedenen Panels einer Anfrage werden mit einem logischen *AND* miteinander verknüpft, während die Elemente innerhalb eines Panels durch ein logisches *OR* zusammenhängen. Jedes Panel kann die folgenden Elemente enthalten: Konzepte aus der ONT-Komponente, zuvor ermittelte Mengen an Patienten sowie Besuchen und andere komplette Anfragen. Für jedes Panel lassen sich unter anderem die folgenden Parameter definieren:

– **panel_timing** gibt an, ob die durch ein Panel gefundenen Elemente nur denselben Patienten oder zusätzlich auch noch im selben Besuch sowie optional außerdem mit derselben *INSTANCENUM* vorkommen müssen.
– Durch **panel_number** werden die Panels beginnend bei 1 nummeriert.
– Mit **panel_date_from** und **panel_date_to** müssen alle gefundenen Fakten innerhalb eines vorgegebenen Zeitraums liegen.
– **invert** wird auf *1* gesetzt, wenn auf den Inhalt des kompletten Panels ein logisches *NOT* angewendet werden soll.
– **total_item_occurrences** erlaubt es, dass nur Patienten beziehungsweise Besuche gefunden werden, in welchen die Elemente des Panels mindestens so oft vorkommen, wie mit diesem Wert angegeben.
– In **item** sind dann die eigentlichen Elemente der Anfrage enthalten. Es enthält im Unterelement **item_key** entweder den Pfad eines Konzepts, die Kennung einer Patiententeilmenge beziehungsweise einer Teilmenge der Besuche, oder die Kennung eines anderen *Query Master*-Objekts. Wenn das Element ein Ontology-Konzept ist, so können noch die folgenden Einschränkungen definiert werden:

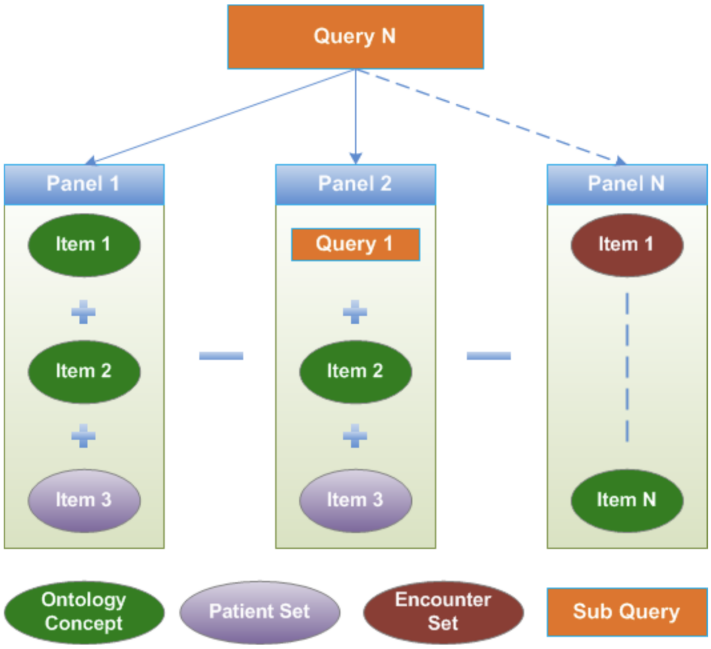

Abb. 4. Aufbau einer CRC-Anfrage (entnommen aus [14])

- **constrain_by_value** erlaubt die Einschränkung auf einen bestimmten Wert oder einen Bereich von Werten. Dazu werden einer oder mehrere Werte zusammen mit einem Operator wie zum Beispiel „gleich“, „zwischen“ oder „größer“ angegeben.
- Mithilfe von **constrain_by_date** kann ein Element für sich auf einen gewissen Zeitraum eingeschränkt werden.
- Durch **constrain_by_modifier** kann außerdem noch der Wert von zugehörigen Modifiern festgelegt werden.

i2b2 unterstützt auch relative, temporale Suchen, bei denen als Teil der Anfrage eine bestimmte zeitliche Beziehung zwischen mehreren Elementen dieser Anfrage festgelegt werden kann. Dazu wird den Elementen, die in Relation zueinander gesetzt werden sollen, in jeweils einem eigenen *subquery*-Element ein Bezeichner zugewiesen. Diese Bezeichner können dann wiederum in einem *subquery_constraint* genutzt werden, um die gewünschten zeitlichen Relationen anzugeben. Eine **Patient Set**-Anfrage beziehungsweise -Antwort besteht aus dem sogenannten *PSMHeader*, gefolgt von der eigentlichen Anfrage (*request*) beziehungsweise Antwort (*response*). Der *PSMHeader* legt dabei die Art der durchgeführten Operation fest. Dies kann unter anderem eine der Folgenden sein:

– **CRC_QRY_getRequestXml_fromQueryMasterId** gibt die XML-Suchanfrage zu einem *Query Master*-Objekt zurück.

- **CRC_QRY_getQueryMasterList_fromUserId** liefert alle bisherigen Anfragen eines Nutzers zurück.
- **CRC_QRY_runQueryInstance_fromQueryDefinition** führt die übergebene Suchanfrage aus. Mit dieser Anfrage wird ein neues *Query Master*-Objekt erzeugt.
- **CRC_QRY_getQueryMasterList_fromGroupId** liefert alle bisherigen Anfragen einer Nutzergruppe zurück.
- **CRC_QRY_getQueryResultInstanceList_fromQueryInstanceId** gibt die Ergebnisse einer Ausführung einer Anfrage zurück.
- **CRC_QRY_getQueryInstanceList_fromQueryMasterId** gibt alle Ausführungen einer Anfrage zurück.
- **CRC_QRY_deleteQueryMaster** löscht ein *Query Master*-Objekt.
- **CRC_QRY_renameQueryMaster** erlaubt die Änderung des Namens eines *Query Master*-Objekts.
- **CRC_QRY_runQueryInstance_fromQueryMasterId** führt ein bestehendes *Query Master*-Objekt erneut aus.
- **CRC_QRY_getResultDocument_fromResultInstanceId** fragt den konkreten Wert eines Ergebnisses einer Ausführung einer Anfrage ab. Das kann zum Beispiel die Anzahl an Patienten sein, auf die eine Anfrage zutrifft.
- **CRC_QRY_runQueryInstance_fromAnalysisDefinition** ermöglicht es ein i2b2-Analyse-Plugin auszuführen. Mit einem dieser Plugins ist es zum Beispiel möglich direkt die Zahl aller Patienten abzufragen, die einen Fakt mit einem vorgegebenen Konzeptpfad haben.
- **CRC_QRY_cancelQuery** erlaubt das Abbrechen einer laufenden Anfrage.
- **CRC_QRY_updateResultInstanceDescription** ermöglicht es die Beschreibung zu ändern, die jede Ergebnisinstanz einer Ausführung einer Anfrage hat.

Das *request*- sowie das *response*-Objekt haben jeweils auch noch ein zur Anfrage passendes *Type*-Attribut. Im *response* ist außerdem immer ein Status-Objekt enthalten, dass zum Beispiel angibt, ob eine Anfrage erfolgreich ausgeführt werden konnte, ob sie abgebrochen wurde, oder ob sie noch läuft.
Eine Suchanfrage wird mithilfe eines *query_ definition*-Objekts definiert, welches unter anderem den Namen, eine Beschreibung sowie die einzelnen Panels der Anfrage enthält. Mithilfe sogenannter *subquery*-Objekte können innerhalb der Anfrage weitere Anfragen definiert werden. Das wird für temporale Anfragen benutzt, bei denen die zeitliche Beziehung zwischen mehreren *subqueries* mithilfe eines *subquery_ constraint*-Objekts zusätzlich zur restlichen Anfrage mit angegeben werden kann.
Nach der *query_ definition* folgt die *result_ output_ list*, die alle Ergebnisse festlegt, die von der Anfrage zurückgegeben werden sollen. Hierfür gibt es die folgenden Möglichkeiten:

- **PATIENTSET** fragt die Teilmenge an Patienten ab, auf die die Anfrage zutrifft.

– Mit **PATIENT_ENCOUNTER_SET** kann die Teilmenge an zutreffenden Besuchen abgefragt werden.
– **PATIENT_COUNT_XML** fragt die Anzahl aller passenden Patienten ab.
– **PATIENT_GENDER_COUNT_XML** gibt die Anzahl an passenden Patienten aufgeteilt auf die Geschlechter zurück.
– **PATIENT_AGE_COUNT_XML** fragt die Anzahl passender Patienten aufgeteilt auf verschiedene Altersgruppen ab.
– Mit **PATIENT_VITALSTATUS_COUNT_XML** wird die Anzahl der gefundenen Patienten aufgeteilt darauf, ob sie noch leben oder bereits verstorben sind, zurückgegeben.
– **PATIENT_RACE_COUNT_XML** erlaubt die Aufteilung der Anzahl gefundener Patienten auf deren Hautfarbe.

Neben der oben erläuterten *Patient Set*-Anfrage gibt es auch noch **Patient Data Object**-Anfragen, die unter anderem von den weiter unten noch beschrieben Plugins der Benutzeroberflächen zur Ausgabe konkreter Daten genutzt werden. Auch diese haben neben einem Header-Objekt, welches hier *PDOHeader* heißt, ein *request*- und ein *response*-Objekt. Der Header legt auch hier wieder die Art der durchzuführenden Operation fest, welche unter anderem eine der folgenden sein kann:

– **getPDO_fromInputList** erlaubt das Abfragen konkreter Werte der CRC-Tabellen. Dazu besteht eine solche Anfrage aus den folgenden drei Teilen:
  • Zunächst wird mithilfe des **input_list**-Objekts die „Eingabemenge“ festgelegt. Diese kann entweder eine zuvor durch obige Anfragen ermittelte Teilmenge der Patienten beziehungsweise Besuche sein. Es kann aber auch ein konkrete Liste an Patienten- beziehungsweise Besuchskennungen angegeben oder die Menge aller Patienten beziehungsweise Besuche durchsucht werden.
  • Mithilfe der **filter_list** kann die Rückgabemenge eingeschränkt werden. Dazu werden an dieser Stelle beliebig viele Panel-Objekte nach obiger Definition angegeben.
  • In **output_option** werden die CRC-Tabellen angegeben, deren zur Anfrage passende Werte zurückgegeben werden sollen. Hier kann auch festgelegt werden, ob alle Spalten oder nur ein Teil davon abgefragt wird. Außerdem ist es möglich anzugeben, dass für jeden Patient nur eine Teilmenge der zu einem Konzept passenden Werte zurückgegeben werden sollen. Dies kann zum Beispiel nur der allererste oder der höchste Wert sein.
– **get_observationfact_by_primary_key** fragt einen Eintrag der *OBSERVATION_FACT*-Tabelle anhand deren Primary Key ab.
– **get_patient_by_primary_key** gibt einen Eintrag der *PATIENT_DIMENSION*-Tabelle anhand ihres Primary Key zurück.
– **get_event_by_primary_key** gibt einen zum übergebenen Primary Key passenden Eintrag der *VISIT_DIMENSION*-Tabelle zurück.

– **get_concept_by_primary_key** fragt mithilfe des Primary Key einen Eintrag der *CONCEPT_DIMENSION*-Tabelle ab.
– **get_observer_by_primary_key** gibt anhand des Primary Key einen Eintrag der *OBSERVER_DIMENSION*-Tabelle zurück.

Wie weiter oben schon erläutert, werden auch diese Anfragen wieder als Teil des *<message_body>*-Tag der i2b2-XML-Struktur spezifiziert. Eine beispielhafte Anfrage des Typs *getPDO_fromInputList* ist in Listing 3 zu sehen.
Hierbei fällt auf, dass die Anfrage selbst auch nochmal aus einem Header, welcher lediglich die Art der Anfrage festlegt, und einer eigentlichen Anfrage besteht. In dem *request*-Teil, werden zunächst anhand der Kennung einer zuvor mit einer anderen Anfrage gefundenen Patientenmenge, der Teil der Daten im System festgelegt, von dem überhaupt Daten abgefragt werden sollen. Die abzufragenden Daten selbst sind in diesem Fall nur das Geschlecht der in der festgelegten Menge enthaltenen Patienten, was über den Pfad zu dem entsprechenden Ontology-Konzept angegeben wird. Abschließend wird noch spezifiziert, dass als Ergebnis sämtliche Spalten der passenden Einträge der *PATIENT_DIMENSION*- sowie der *OBSERVATION_FACT*-Tabelle zurückgegeben werden sollen.

```
<pdoheader>
  <request_type>getPDO_fromInputList</request_type>
</pdoheader>
<request type="GetPDOFromInputList_requestType">
  <input_list>
    <patient_list>
      <patient_set_coll_id>2</patient_set_coll_id>
    </patient_list>
  </input_list>
  <filter_list>
    <panel>
      <panel_number>1</panel_number>
      <item>
        <item_key>\\i2b2_DEMO\i2b2\Demographics\Gender\</item_key>
      </item>
    </panel>
  </filter_list>
  <output_option>
    <patient_set select="using_input_list" onlykeys="false"/>
    <observation_set blob="true" onlykeys="false"/>
  </output_option>
</request>
```

Listing 3. Beispielhafte *getPDO_fromInputList*-Anfrage

2.2.1.4 Weitere Hive-Zellen

Neben den beiden oben beschriebenen i2b2-Zellen, welche für die zentrale Funktionalität von i2b2 zuständig sind, gibt es noch einige weitere Komponenten, die zum sogenannten Hive gehören und im Folgenden genauer beschrieben werden.

Project Management (PM) Cell

Die Erläuterungen zur PM-Zelle basieren auf [16] und [17]. Diese Zelle hat zwei zentrale Aufgaben. Einerseits steuert sie den Zugriff der Nutzer auf die verschiedenen i2b2-Dienste und andererseits speichert sie die Adressen unter denen diese Dienste erreichbar sind. Aufgrund dieser Funktionalität wird diese Zelle auch als Administrationsmodul von i2b2 bezeichnet.
Für die Anmeldung in einem i2b2 System werden zunächst die Rollen eines Nutzers überprüft. Jeder Nutzer kann dabei eine oder mehrere der folgenden Rollen haben:

- **USER:** Diese Rolle haben in der Regel „normale“ Mitglieder eines Forschungsteams. Sie können die medizinischen Daten in einem Projekt (Projekte werden weiter unten noch erläutert) abfragen sowie Anpassungen an ihrem eigenen Nutzerprofil vornehmen.
- **MANAGER:** Ein Nutzer mit dieser Rolle verwaltet ein komplettes Forschungsteam. Er kann mit seinem Projekt verbundene *USER* bearbeiten sowie neue hinzufügen. Außerdem kann er Metadaten des Projekts verändern sowie neue Metadaten hinzufügen.
- **ADMIN:** Ein solcher Nutzer muss nicht Teil eines Forschungsteams sein und ist auch keinem Projekt zugeordnet. Er ist für alle administrativen Aufgaben in Verbindung mit i2b2 zuständig, weswegen er die Daten aller Nutzer und Projekte anpassen sowie neue Benutzer und Projekte anlegen kann. Auch kann er Daten des i2b2-Hive konfigurieren und neue hinzufügen, was im Folgenden noch genauer erläutert wird.

Die PM-Zelle definiert einen oder mehrere i2b2-Hives. Jeder Hive hat einen eindeutigen Domain-Namen, der bei der Anmeldung mit angegeben werden muss. In einem Hive sind dann die Adressen der einzelnen i2b2-Komponenten sowie optional zusätzliche Konfigurationsparameter der Komponenten gespeichert.
Innerhalb eines Hives gibt es dann ein oder mehrere Projekte. Jeder Nutzer (außer denen mit der Rolle *ADMIN*) ist einem oder mehreren Projekten zugeordnet. Ist ein Nutzer mehreren Projekten zugeordnet, so muss er nach der Anmeldung an einem Hive das gewünschte Projekt auswählen. Für jedes Projekt werden in der PM-Zelle neben optionalen Konfigurationsparametern die Nutzer mit ihrer Rolle (*USER* oder *MANAGER*) in dem Projekt gespeichert. Außerdem werden für jeden Nutzer noch eine oder mehrere der folgenden **Data Protection role**s gespeichert.

- **DATA_OBFSC:** (OBFSC = Obfuscated) Ein solcher Nutzer kann nur aggregierte Daten, die außerdem „verschleiert“ sind, wie zum Beispiel die Anzahl an zu einer Anfrage passenden Patienten, einsehen. Außerdem kann er dieselbe Anfrage in einem bestimmten Zeitraum nur eine festgelegte Anzahl mal ausführen. Falls diese Limitierung überschritten wird, so wird der

Account gesperrt und muss erst von einem Administrator wieder entsperrt werden.
- **DATA_AGG:** (AGG = Aggregated) Dieser Nutzer kann nur aggregierte Daten einsehen, die allerdings nicht „verschleiert“ werden. Auch ist die Anzahl der Ausführungen einer Anfrage für diesen Nutzer nicht limitiert.
- **DATA_LDS:** (LDS = Limited Data Set) Ein Nutzer mit dieser Rolle kann alle CRC-Felder mit Ausnahme der Verschlüsselten einsehen. Beispiele für möglicherweise verschlüsselte Felder sind die *BLOB*-Felder der CRC-Tabellen.
- **DATA_DEID:** (DEID = De-identified Data) Ein solcher Nutzer kann sämtliche CRC-Felder inklusive der Verschlüsselten einsehen.
- **DATA_PROT:** (PROT = Protected) Dieser Nutzer kann sämtliche Daten inklusive der Daten, die nicht anonymisiert sind, in der *Identity Management*-Zelle (wird weiter unten erläutert) einsehen.

Die zentrale Anfrage an diese Zelle lautet **getUserConfiguration**, mit der ein Nutzer angemeldet und alle notwendigen Parameter des Nutzers, des gewünschten Hives sowie des ausgewählten Projekts zurückgeliefert werden. Neben dieser Anfrage gibt es noch viele weitere, mit denen die i2b2-Administration ermöglicht wird.

Workplace Framework (WORK) Cell

Die folgenden Ausführungen zu dieser i2b2-Komponente basieren auf [18] und [19]. Der Workplace in i2b2 enthält unter anderem die Konzepte und Anfragen, die ein Nutzer am häufigsten benötigt und wird so zu dessen persönlichem Arbeitsplatz. Diese Komponente bietet die folgenden drei Möglichkeiten:

- Sie speichert den Workplace jedes Nutzers und bietet diesem die Möglichkeit ihn zu verwalten.
- Sie erlaubt es, Elemente des Workplace eines Nutzers mit anderen Nutzern im selben Projekt zu teilen.
- Der Team-Manager hat die Möglichkeit sich den Workplace aller Projektteilnehmer anzusehen.

Sowohl Nutzer mit der Rolle *USER* als auch Nutzer mit der Rolle *MANAGER* haben die Möglichkeit eigene Workplaces anzulegen und zu organisieren. Außerdem können sie Teile ihrer Workplaces mit anderen Projektnutzern teilen sowie Informationen ansehen, die von anderen Nutzern mit ihnen geteilt wurden.
Die folgenden Elemente können alle in einem Workplace gespeichert werden:

- Konzepte aus der Ontology-Komponente
- Durch Suchanfragen erhaltene Teilmengen aller Patienten beziehungsweise Besuche
- Frühere Suchanfragen inklusive ihrer Ausführungen und Ergebnisse
- Die Definition einer Suchanfrage
- Einzelne Panels einer Suchanfrage
- Einzelne abgefragte CRC-Fakten

– Einzelne Patienten oder Besuche aus den oben beschriebenen Teilmengen
– Einzelne XML-Ergebnisse einer Suchanfrage (zum Beispiel die Anzahl an passenden Patienten)

All diese Elemente können in mehreren Ordnern innerhalb der Workplaces organisiert werden. Auch kann jedes der Elemente in eine XML-Datei exportiert werden.
Über XML-Nachrichten ist auch bei dieser i2b2-Komponente der Zugriff auf alle obigen Funktionen möglich.

Identity Management Framework (IM) Cell
Die nun folgenden Ausführungen zur IM-Zelle basieren auf [20] und [21]. Diese Komponente dient dazu, die Herkunft eines Patienten, also dessen Kennung in dem System, aus dem seine Daten in i2b2 importiert wurden, zu speichern. Da über diese Kennung ein Patient eindeutig einer realen Person zugeordnet werden kann, kann diese Zuordnung in der IM-Zelle verschlüsselt abgelegt werden. So können einerseits Auswertungen auf den medizinischen Daten durchgeführt werden, ohne Datenschutzprobleme zu bekommen und andererseits wenn zum Beispiel mit i2b2 Patienten für Studien gesucht werden, deren i2b2-Kennungen auch wieder den zugehörigen Personen zugeordnet werden. Nur Benutzer mit der „Data Protection Role“ $DATA\_PROT$ (siehe oben) können auf diese Daten zugreifen.
Die folgenden Daten werden in der IM-Komponente gespeichert:

– Die medizinischen Einrichtungen (genannt *Sites*), aus denen die Daten stammen.
– Die Projekte der verschiedenen Einrichtungen.
– Die Patienten in den einzelnen Projekten.
– Demographische Daten zu den Patienten.
– Die Zuordnung der Patientenkennung im ursprünglichen System zu der in i2b2.
– Alle Zugriffe auf Patientendaten über die IM-Zelle.

Diese Komponente erlaubt es auch, denselben Patienten, wenn dieser in mehreren Quellsystemen (auch mit unterschiedlichen Kennungen) vorhanden ist, einer einzelnen i2b2-Kennung zuzuordnen um so auch Auswertungen über die Daten einer einzelnen Einrichtung hinweg durchführen zu können.
Projektmanager und Administratoren haben zusätzlich zum Einsehen der Zuordnungen der verschiedenen Kennungen noch die Möglichkeit abzufragen, welche Zugriffe es auf diese Zuordnungen gab. Dies können entweder alle Zugriffe eines bestimmten Nutzers, alle Zugriffe auf einen bestimmten Patienten, oder alle Zugriffe innerhalb eines Projektes sein.
Die XML-Schnittstelle der IM-Komponente kann außer zum Abfragen der beschriebenen Zugriffe noch für folgendes genutzt werden:

– Der AES-Schlüssel, mit dem die Daten wieder entschlüsselt werden können, wird immer nur im Arbeitsspeicher gespeichert und muss somit nach jedem

Starten der IM-Zelle für jedes einzelne Projekt wieder neu angegeben werden. Dies und auch eine Überprüfung, ob der Schlüssel schon im Arbeitsspeicher liegt, ist über die XML-Schnittstelle möglich.
- Bei dem Abfragen eines *Patient Data Object*s mit der entsprechenden CRC-Anfrage wird, wenn konkrete Kennungen aus anderen Systemen zur Einschränkung der Ergebnisse angegeben wurden, die IM-Zelle im Hintergrund abgefragt, um diese Kennungen eventuell zu entschlüsseln und ihnen die entsprechende i2b2-Kennung zuzuordnen.
- Es kann überprüft werden, ob eine oder mehrere Kennungen aus verschiedenen medizinische Einrichtungen einem bestimmten i2b2-Projekt zugeordnet wurden.

File Repository (FR) Cell

Die folgenden Erläuterungen zur FR-Komponenten basieren auf [22] und [23]. Sie hat zwei Grundlegende Aufgaben:

- Dateien und zugehörige Metadaten verwalten
- Dateien zu speichern und abzurufen

Die XML-Schnittstelle dieser Komponente bietet die Möglichkeit, eine neue Datei hochzuladen und zu speichern sowie eine bestehende Datei wieder herunterzuladen.
Diese i2b2-Zelle wird in der Regel einerseits dafür benutzt, neue Patientendaten, die im XML-Format übermittelt werden, hochzuladen, um sie dann mit den anderen Komponenten weiter verarbeiten zu können. Anderseits können mit ihr ganz allgemein bestehende Dateien verwaltet und neue hinzugefügt werden. So kann auf diese Dateien von anderen i2b2-Zellen aus zugegriffen oder Verweise darauf gespeichert werden, um zum Beispiel einem Patienten die Datei eines Arztbriefes zuzuordnen.

Natural Language Processing (NLP) Cell

Die Erläuterungen zu dieser Zelle basieren auf [24] und [25]. Mit der NLP-Komponente ist es möglich aus unstrukturierten Textdokumenten klinische Daten zu extrahieren. Die folgenden Extraktionen werden zur Zeit unterstützt:

- **Principal diagnoses extraction:** Hierbei werden sogenannte Hauptdiagnosen extrahiert. Damit ist die Diagnose gemeint, die der primäre Grund für die Aufnahme in eine medizinische Einrichtung war (vergleiche [26]). Die Extraktion liefert unter anderem die zugehörigen Konzepte, den Begriff im Dokument, der dem Konzept zugeordnet wurde, den Titel des Dokumentenabschnitts, in dem die Diagnose entdeckt wurde und eine Liste von gefundenen Modifiern zu der Diagnose. Solche Modfifier können zum Beispiel Negationen oder die Familien-Krankheitsgeschichte sein.
- **Discharge medications extraction:** Hiermit können alle Konzepte extrahiert werden, die sich auf die Medikation eines Patienten bei dessen Entlassung aus einer medizinischen Einrichtung beziehen. Auch hier wird zusätzlich der Titel des Dokumentenabschnitts zurückgegeben, in dem das Konzept gefunden wurde.

- **Smoking status extraction:** Mit dieser Extraktion kann herausgefunden werden, ob und wie viel der in einem Dokument beschriebene Patient raucht. Diese Information wird in Verbindung mit dem Titel des Abschnitts zurückgegeben, in dem sie gefunden wurde.
- **Extraction of all available concepts:** Diese Extraktion erlaubt es sämtliche in einem Dokument gefundene und von der NLP-Zelle unterstütze Konzepte zu erhalten.
- **Custom concepts extraction:** Hiermit ist es möglich eigene Konzept-Extraktionen auf ein Dokument anzuwenden. Diese müssen in einem vorgegebenen XML-Format beschrieben werden, welches wiederum auf die Klassen verweist, die die Extraktion durchführen.

Für den kompletten Ablauf der Sprachverarbeitung ist das **GATE NLP Framework** (im Folgenden beschrieben anhand von [27]) zuständig. Dies ist eine Open Source Lösung für praktisch jede beliebige Form von Sprachverarbeitung. Der grobe Verarbeitungsablauf dieses Frameworks sieht wie folgt aus:

- Zunächst wird eine Menge beliebiger unstrukturierter Texte festgelegt. Im Falle von i2b2 sind das dann beispielsweise mehrere Arztbriefe.
- Anschließend muss die zu verwendende Terminologie in strukturierter Form angegeben werden. Damit sind die Informationen gemeint, deren zugehörige Werte aus den obigen Texten extrahiert werden sollen. Für die zuvor beschriebene *Principal diagnoses extraction* sind dies beispielsweise Daten im UMLS-Format („Unified Medical Language System“), wie sie von [28] bezogen werden können.
- Dann wird ein sogenannter „Gold Standard“ benötigt, der einen Teil der Texte mit der Terminologie in Verbindung bringt. Diese Zuordnungen stehen im Falle der *Principal diagnoses extraction* ebenfalls in der UMLS-Datenbank.
- Als nächstes wird eine *Pipeline* definiert, die dann für die automatische Verarbeitung der restlichen Texte mithilfe des Gold Standards zuständig ist, sowie diese Verarbeitung gestartet.
- Als Rückgabe erhält man abschließend die Texte, die um passende Annotationen anhand der Terminologie ergänzt wurden. i2b2 wandelt diese dann in entsprechende XML-Dokumente um und gibt sie zurück.
- GATE bietet außerdem noch verschiedene Möglichkeiten zum Durchsuchen der annotierten Texte, was von der i2b2-NLP-Komponente jedoch nicht genutzt wird.

Diese Zelle von i2b2 verfügt außerdem noch über ein einen „N-Gram Finder“, mit dem die Häufigkeit bestimmter Zeichenketten und Wörter aus den Texten extrahiert werden kann.
Speziell für die *Smoking status extraction* kommt zusätzlich noch die „WEKA Data Mining Library“ (vergleiche [29]) zum Einsatz. Dies ist eine frei verfügbare Data-Mining-Software, die im Rahmen dieser Arbeit allerdings nicht genauer beschrieben wird.

Pulmonary Function Test Processing (PFT) Cell
Die folgenden Erklärungen basieren auf [30]. Diese i2b2-Komponente ermöglicht es Berichte von Lungenfunktionstests zu verarbeiten und die gemessenen Werte in Form von i2b2-CRC-Fakten zu extrahieren. Hierzu müssen die Berichte in einem vorgegebenen Format vorliegen.

2.2.2 Webclient und Workbench

In diesem Abschnitt werden die für i2b2 zur Verfügung gestellten graphischen Oberflächen beschrieben. Die folgenden Ausführungen basieren direkt auf den Oberflächen, die von [7] heruntergeladen werden können sowie den darin enthaltenen Erläuterungen.
In beiden Tools können mithilfe einer Konfigurationsdatei mehrere i2b2-Domains mit ihrer jeweiligen Adresse konfiguriert werden, zwischen denen man dann beim Starten der Programme auswählen kann. An der ausgewählten Domain kann man sich dann anmelden und bekommt bei erfolgreicher Anmeldung die Startoberfläche zu sehen. Der sogenannte Webclient ist in Abbildung 5 und die Workbench in Abbildung 6 zu sehen. Beide bieten im oberen linken Bereich die Möglichkeit,

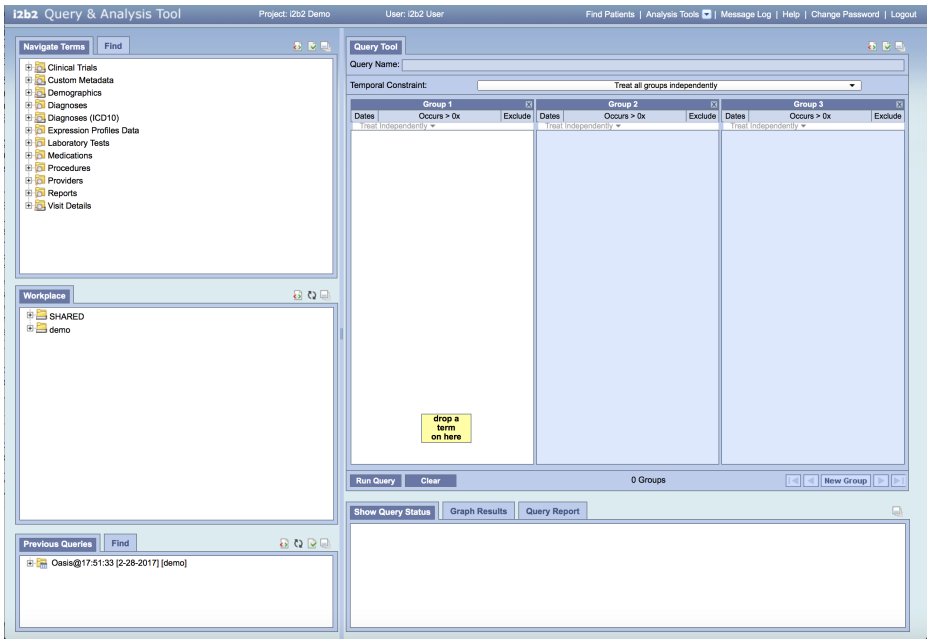

Abb. 5. Oberfläche des i2b2 Webclient (eigener Screenshot)

den Inhalt der Ontology-Komponente darzustellen. Hierbei wird jeweils zunächst die oberste Hierarchieebene angezeigt, die sich - solange es noch untergeordnete Objekte gibt - immer weiter aufklappen lässt. Hier bieten beide Oberflächen

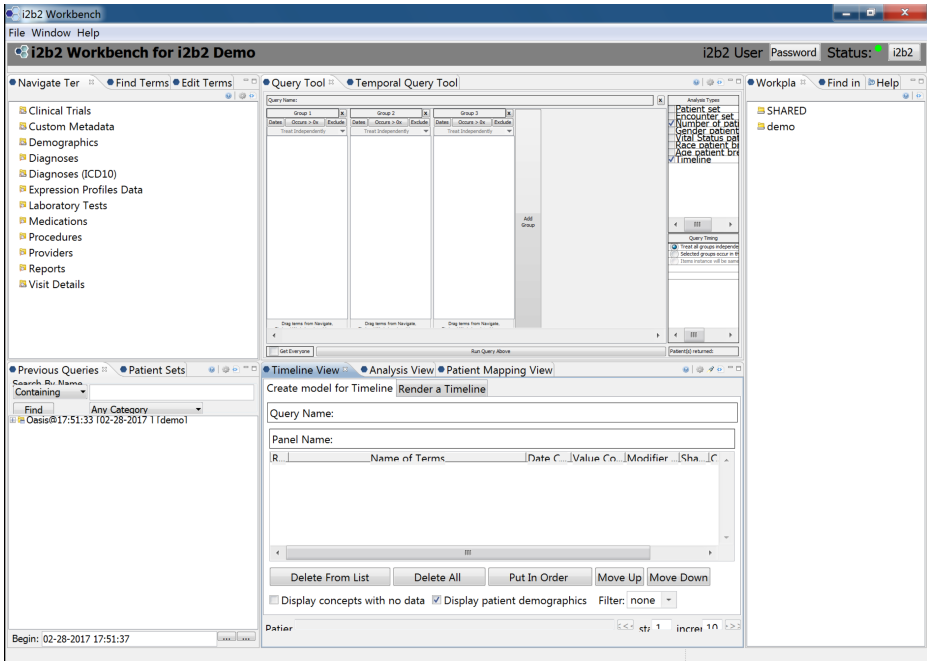

Abb. 6. Oberfläche der i2b2 Workbench (eigener Screenshot)

auch die Möglichkeit, die gespeicherten Konzepte zu durchsuchen, wobei die Suche darauf beschränkt werden kann, nur die Elemente unter einem bestimmten Konzept auf höchster Hierarchieebene zu durchsuchen. Auch können hier die verschiedenen Codierungssysteme ausgewählt werden, um dann nach einem Konzept mit einem speziellen Basecode zu suchen. In diesem Teil der Oberfläche kann außerdem unter anderem eingestellt werden, wie viele Konzepte unter einem Elternknoten maximal dargestellt werden sollen, ob Synonyme, versteckte Konzepte, Modifier, sowie die Zahl der zu einem Konzept passenden Patienten angezeigt werden sollen. Die i2b2 Workbench erlaubt hier außerdem das Anpassen bestehender sowie das Hinzufügen neuer Konzepte, sofern diese als editierbar markiert sind. Auch kann über die Workbench an dieser Stelle eine Synchronisation der ONT-Zelle mit der Konzept-Tabelle der CRC-Zelle gestartet werden sowie angezeigt werden, ob eine solche Synchronisation überhaupt notwendig ist. Aus dieser Ansicht können die einzelnen Konzepte dann in die anderen Teile der Oberfläche gezogen werden.
Links unten zeigen beide Oberflächen eine einstellbare Zahl an vorherigen Suchanfragen. Diese lassen sich aufklappen, um so die einzelnen Ausführungen sowie die Ergebnisse der jeweiligen Ausführungen ansehen zu können. Auch die vorherigen Anfragen lassen sich durchsuchen. Außerdem lässt sich einstellen, wie die Anfragen sortiert sein sollen. Die i2b2 Workbench ermöglicht zusätzlich noch das Einschränken der Anfragen auf Anfragen ab einem bestimmten Zeitpunkt. Auch

zeigt die Workbench eine spezielle Ansicht, in welcher nur die bisher gefundenen Teilmengen aller Patienten dargestellt werden. Aus dieser Ansicht lässt sich außerdem die Definition einer vorherigen Anfrage erneut in der Oberfläche darstellen sowie komplette vorherige Anfragen und Ergebnisse davon als Teil einer neuen Anfrage verwenden.
Links in der Mitte im Webclient sowie am rechten Rand der Workbench befindet sich der sogenannte Workplace, in dem sich Nutzer häufig verwendete Anfragen sowie Bestandteile von Anfragen speichern und mit anderen Nutzern teilen können. In der Workbench kann der Workplace auch direkt durchsucht werden, was im Webclient über ein spezielles Plugin ebenfalls möglich ist.
Zentral in beiden Oberflächen befindet sich die Möglichkeit, die eigentlichen Suchanfragen zusammenzubauen. Dazu können die einzelnen Panels der Suche definiert, zeitlich relative Anfragen erstellt, die Anfrage benannt sowie die gewünschten Suchergebnisse ausgewählt werden. Dabei ist es außerdem möglich die Werte sowie Zeiträume von Konzepten einzuschränken, anzugeben, wie oft ein Konzept mindestens bei einem Patienten vorkommen soll, ein komplettes Panel zu negieren, neue Panels hinzuzufügen und mehrere sogenannte Events zusammen mit den zeitlichen Beziehungen zwischen ihnen anzugeben. Allgemein einstellen lässt sich in diesem Teil der Oberfläche, wie lange auf die Antwort einer Suchanfrage gewartet werden soll.
Unter der Definition der Suchanfrage werden dann deren Ergebnisse angezeigt. Im Webclient ist hier standardmäßig eine grafische Darstellung der ausgewählten XML-Ergebnisse (vergleiche die Erläuterungen zur CRC-Komponente) zu sehen. Daneben zeigt dieser in einer eigenen Ansicht noch eine kompakte Darstellung aller Ergebnisse sowie in einer weiteren Ansicht einen kompletten Bericht zur ausgeführten Anfrage, der die Anfrage nochmal in Worten und graphisch darstellt sowie alle Ergebnisse - sofern möglich - sowohl als Diagramm als auch als Tabelle zeigt. Dieser Bericht lässt sich auch ausdrucken. Die Workbench zeigt hier standardmäßig eine Timeline aller gefundenen Fakten, also wann für welchen Patient welches der gesuchten Konzepte in einem Fakt gefunden wurde. Zum Anzeigen der XML-Ergebnisse bietet die Workbench den sogenannten „Analysis View“, der diese als Diagramm darstellt. Im selben Teil der Oberfläche befindet sich bei der Workbench außerdem noch der „Patient Mapping View“, mit dem die Zuordnung der i2b2-Patientenkennungen zu den Kennungen in den Systemen, aus dem die Daten importiert wurden, angezeigt werden können.
Beide Oberflächen bieten außerdem noch die Möglichkeit, das Passwort des Nutzers zu verändern. Im Webclient können die einzelnen Ansichten vergrößert werden, während sie in der Workbench auch beliebig neu angeordnet, bildschirmfüllend dargestellt und komplett ausgeblendet werden können. Beide bieten auch eine eigene Hilfefunktion zur Erklärung der einzelnen Funktionen und eine Möglichkeit die bisher im Hintergrund gesendeten XML-Anfragen sowie die darauf erhaltenen XML-Antworten zu betrachten.
Zusätzlich zu den obigen Möglichkeiten bietet der Webclient noch die folgenden sogenannten „Analysis Tools“:

– Mit **Demographics** kann eine gefundene Menge von Patienten nach ihren gespeicherten demographischen Werten aufgeteilt in mehreren Diagramme dargestellt werden. So sieht man zum Beispiel, wie viele Patienten in unterschiedlichen Altersgruppen gefunden wurden. Mit diesem Tool lassen sich auch die demographischen Werte zweier gefundener „Patient Sets“ miteinander vergleichen.
– **Timeline** erlaubt die selbe zeitliche Darstellung verschiedener Fakten, die auch oben für die Workbench beschrieben wurde.
– Der **Workplace Items Sharing Enhancement - Searcher** erlaubt auch für den Webclient das Durchsuchen des Workplace eines Nutzers. Hierbei können außerdem gewisse Begriffe ausgeschlossen werden, sodass diese nicht in den gefundenen Elementen enthalten sind.
– Mithilfe des **Cohort Analysis & Refinement Expeditor** können die demographischen Daten eines gefundenes „Patient Sets“ auf mehrere unterschiedliche Konzepte aufgeteilt als Diagramme angezeigt werden. Auch ist es mit diesem Tool möglich anzuzeigen, wie oft ein Konzept pro Patient vorkommt, was dann auch wieder auf die einzelnen demographischen Gruppen aufgeteilt wird.
– **ExportXLS** ermöglicht es sich die Werte bestimmter Konzepte einer gefundenen Menge von Patienten ausgeben und exportieren zu lassen.

Die Workbench bietet außer den oben beschriebenen Möglichkeiten noch das sogenannte „Managers Tool“, das die folgenden Funktionen bietet:

– Es kann die in der ONT-Komponente gespeicherte Anzahl an passenden Patienten zu jedem Konzept automatisch aktualisiert werden.
– Der zur Entschlüsslung der Werte in der IM-Komponente nötige AES-Schlüssel kann gesetzt werden. Auch ist es möglich zu überprüfen, ob bereits ein Schlüssel angegeben wurde.
– Zusätzlich können noch die Zugriffe auf die Werte der IM-Komponente angezeigt und nach Nutzer, medizinischer Einrichtung sowie i2b2-Projekt gefiltert werden.

2.2.3 Plugins

Neben den oben erläuterten Komponenten bietet i2b2 noch eine Reihe von Plugins, die im Folgenden beschrieben werden. Alle diese Plugins sind optional und somit nicht zwingend für die Funktionalität von i2b2 erforderlich.

Patient Counts Plugin

Dieses Plugin, wozu die Erläuterungen auf [31] basieren, ist das einzige angebotene Plugin für die CRC-Komponente von i2b2. Es wird mithilfe der oben beschriebenen CRC-Anfrage *CRC_QRY_runQueryInstance_fromAnalysisDefinition* ausgeführt und erhält als einzigen Parameter den vollständigen Pfad zu einem Konzept der Ontology-Komponente. Anhand davon berechnet es, bei wie vielen Patienten dieses Konzept oder eines der Konzepte, die in der Hierarchie unter dem Angegebenen hängen, vorkommen. Diese Zahl liefert es dann zurück.

High Performance Computing Plugin
Die folgenden Beschreibung des HPC-Plugins basiert auf [32]. Mithilfe dieses Plugins wird es den Nutzern von i2b2 ermöglicht, direkt auf „entfernte Berechnungsressourcen“ (in der Regel Cluster von Computern) zuzugreifen. Hierzu können über eine XML-Syntax zum Beispiel Skriptdateien mittels FTP an das Cluster übertragen und dort mit beliebigen Parametern ausgeführt werden. Anschließend kann die lokale i2b2-Installation sogar komplett beendet werden und zu einem beliebigen späteren Zeitpunkt der Status der Skriptausführung abgefragt werden. Für all diese Funktion bietet das Plugin auch eine eigene Ansicht zum Steuern über die i2b2 Workbench, aus welcher sich die Ergebnisse der Ausführung auch in einer lokalen Datei speichern lassen.

Text Analyzer Plugin
Dieses Plugin wird im Folgenden anhand von [33] beschrieben. Es erlaubt aus der Timeline-Ansicht der Workbench heraus den Inhalt der *OBSERVATION_BLOB*-Spalte eines gefunden CRC-Fakts mit einer *valTypeCd* von *B* anzusehen. Dies sind in der Regel komplette Texte wie zum Beispiel Arztbriefe oder Laborberichte. Wenn diese verschlüsselt vorliegen, so fragt das Plugin vor dem Anzeigen auch das Passwort zur Entschlüsselung ab. Im geöffneten Text ist es möglich einzelne Konzepte hervorzuheben. Dazu können dieses aus der Ontology-Ansicht der Workbench in die NLP-Ansicht dieser Komponente gezogen werden, worauf hin die gewünschten Konzepte mithilfe der i2b2-NLP-Komponente (siehe oben) gesucht und markiert werden. Außerdem erlaubt dieses Plugin auch das durchsuchen des Textes nach beliebigen Worten.

Table View Plugin und Export Data Plugin
Die folgenden Erläuterungen zu diesen Plugins basieren auf der HTML-Dokumentation der Plugins, die bei dem von [7] herunterladbaren Quellcode der Plugins enthalten sind. Da die Dokumentation beider Plugins identisch ist, werden sie hier in einem Punkt beschrieben. In der Ansicht, die die Plugins der Workbench hinzufügen, können zunächst mehrere durch frühere Suchanfragen ermittelte Mengen von Patienten angegeben werden. Auch müssen die Konzepte inklusive optionaler Einschränkungen des betrachteten Zeitraums sowie Wertebereichs festgelegt werden, deren zugehörige Fakten in der Ausgabe enthalten sein sollen. Anschließend präsentieren diese Plugins die demographischen Daten der Patienten sowie die gewünschten Fakten in tabellarischer Form sowie falls möglich auch als Diagramm. Diese Ergebnisse können dann auch exportiert sowie frühere Ergebnisse zu deren Anzeige wieder importiert werden.

Correlation Analysis Plugin
Dieses Plugin wird anhand von [34] erläutert. Es bietet eine weitere zusätzliche Ansicht für die Workbench, mit welcher sich zunächst mehrere Konzepte und entweder alle Patienten oder Teilmengen davon auswählen lassen. Anhand dieser Daten wird anschließend der Zusammenhang zwischen den Fakten der einzelnen Konzepte berechnet und in tabellarischer Form dargestellt (die genauen Berechnungen dahinter werden ebenfalls in [34] beschrieben). Das Ergebnis

dieser Berechnung lässt sich zusammen mit den gewählten Fakten und Patienten auch exportieren sowie wieder importieren.

Import Data Plugin
Die folgenden Erklärungen basieren auf [35]. Dieses Plugin erweitert die i2b2 Workbench um eine Ansicht, mit der direkt Fakten, Patienten sowie Besuche in i2b2 importiert werden können. Zu diesem Zweck kann zunächst eine CSV-Datei ausgewählt werden, die importiert werden soll. Anschließend wird angegeben, in welcher Zeile der Header der Datei steht, mit welchem Zeichen die einzelnen Spalten getrennt sind, sowie welches Zeichen zum Beenden einer Zeilen und zum Markieren von zusammenhängendem Text verwendet wird. Anschließend muss ausgewählt werden, ob es sich beim Inhalt der aktuellen Datei um Fakten, Patienten oder Besuche handelt, woraufhin die Spalten der Datei den Spalten der CRC-Tabelle zugeordnet werden müssen. Sobald dies für alle zu importierenden Dateien wiederholt wurde, können die Daten in tabellarischer Form angezeigt und auf Fehler überprüft werden. Bei vorhandenen Fehlern muss die Datei entsprechend korrigiert und die obigen Schritte wiederholt werden. Vor dem eigentlichen Importieren kann dann noch eingestellt werden, ob die übermittelten Kennungen für Patienten und Besuche sowie die Texte verschlüsselt werden sollen. Auch kann angegeben werden, ob für fehlende Patienten- und Besuchskennungen zufällige IDs generiert werden sollen. Außerdem lässt sich noch festlegen, wie viele Einträge der CSV-Datei auf einmal importiert werden sollen. Nach dem eigentlichen Starten des Imports kann dann noch dessen Status sowie der Status früherer Importe überprüft werden.

Annotator Plugin
Dieses Plugin wird im Folgenden anhand von [36] beschrieben. Zunächst wird eine Menge an Patienten sowie mehrere Konzepte vom Typ „DOC“ angegeben. Anschließend werden mindestens zwei Konzepte ausgewählt (zum Beispiel „positives Vorkommen von Fieber“ und „negiertes Vorkommen von Fieber“), nach denen die Dokumente, die zu den zuvor ausgewählten Konzepten und Patienten gefunden wurden, durchsucht werden sollen. Außerdem kann ein Konzept angegeben werden (zum Beispiel „keine Erwähnung von Fieber“), welchem Dokumente zugeordnet werden sollen, in denen die gesuchten Konzepte überhaupt nicht vorkommen. Um diese Dokumente zu bestimmen können Begriffe festgelegt werden, die auf jeden Fall vorkommen müssen, wodurch nur dazu passende Dokumente in den folgenden Schritten genauer betrachtet werden. Anschließend müssen sogenannte „Features“ angegeben werden. Dies sind reguläre Ausdrücke, die den gewählten Konzepten zugeordnet werden. Diese können entweder selber definiert oder aus einer XML-Datei importiert werden. Auch besteht die Möglichkeit sich die gefundenen Dokumente direkt anzeigen zu lassen und die Vorkommen der Konzepte selber auszuwählen, woraus dann automatisch ein regulärer Ausdruck generiert wird. Nachdem so die ersten Dokumente den Konzepten zugeordnet wurden kann der nächste Satz an Dokumenten geladen werden. Dies kann entweder rein zufällig gemacht werden, oder es werden die bereits vorhandenen Features zum Klassifizieren der neuen Dokumente verwendet um dem

Nutzer nur die Dokumente zur manuellen Klassifizierung anzuzeigen, bei denen sich der automatische Klassifizierer am unsichersten ist. Das kann dann solange wiederholt werden, bis die Trefferquote der Klassifizierung einen gewünschten Wert erreicht. Daraufhin können die restlichen Dokumente automatisch den Konzepten zugeordnet und das Ergebnis dieser Zuordnung in einer Datei gespeichert werden. Der Zweck dieses Plugins ist es, Nutzern, die sich mit Sprachverarbeitung nicht oder kaum auskennen, einen vergleichsweise einfachen Zugang dazu zu ermöglichen.

2.3 i2b2 Wizard

Mit dem i2b2 Wizard, welcher im Folgenden anhand von [37] genauer beschrieben wird, ist es möglich, eine i2b2-Installation in einer Linux-Umgebung einzurichten, zu konfigurieren und zu verwalten, was alles auch im Rahmen dieser Arbeit für die verwendete Testinstallation von i2b2 genutzt wurde. Der Hauptoberfläche des i2b2 Wizard ist in Abbildung 7 zu sehen.

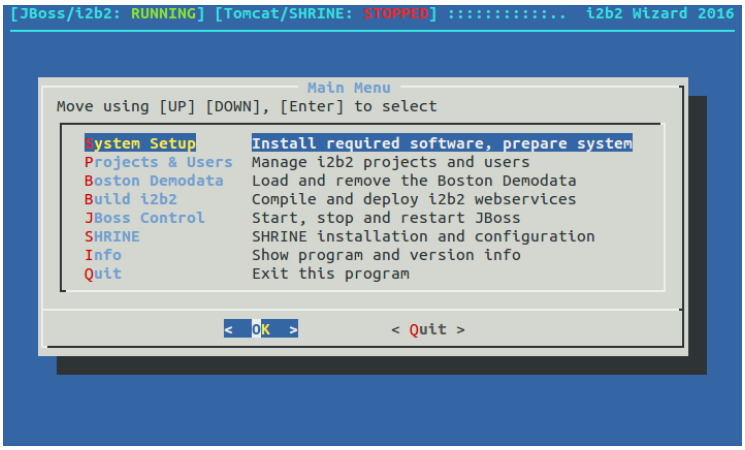

Abb. 7. Oberfläche des i2b2 Wizard (eigener Screenshot)

Konkret bietet der i2b2 Wizard die folgenden Funktionen:

- Alle Funktionen des i2b2 Wizard stehen über ein Kommandozeilentool mit größtenteils graphischer Benutzeroberfläche (die allerdings ausschließlich über die Tastatur bedient wird) zur Verfügung.
- Er ermöglicht die komplette Installation von i2b2 inklusive der folgenden Schritte:
  - Die benötigten Programme werden automatisch heruntergeladen und extrahiert.
  - Die Programme, die im genutzten Linux-System installiert sein müssen, werden automatisch installiert.

    - Der i2b2-Quellcode wird automatisch konfiguriert und kompiliert.
    - Die Datenbank im Hintergrund wird automatisch und inklusive der benötigten Tabellen angelegt.
- Es ist außerdem möglich, die folgenden i2b2-Administrationsaufgaben durchzuführen:
    - i2b2-Projekte und i2b2-Nutzer können angelegt und gelöscht werden.
    - Die Rollen und somit die Rechte der einzelnen i2b2-Nutzer können konfiguriert und die Nutzer einem Projekt zugeordnet werden.
    - Der für i2b2-Installationen zur Verfügung gestellte Satz an Beispieldaten (genannt „Boston Demodata“) kann in die i2b2-Installation geladen und aus dieser wieder gelöscht werden.
    - Die sogenannte Hive-ID, welche den Namen eines i2b2-Servers darstellt, kann verändert werden.
    - Die Netzwerkschnittstelle, über die i2b2 von außerhalb des Rechners erreichbar sein soll, kann verändert werden.
    - Es kann der entfernte Zugang auf die verwendete Datenbank aktiviert werden.
    - Die verwendeten Passwörter können geändert werden.
- Falls für eine Aufgabe während der Installation oder der Administration ein Programm fehlt, so wird versucht, dieses automatisch zu installieren.
- Der i2b2 Wizard ermöglicht auch die Installation und Konfiguration des „Shared Health Research Information Network“ (SHRINE). Dies stellt eine Möglichkeit dar, mehrere i2b2-Installationen zu verbinden, eine Suchanfrage an sämtliche dieser Installationen zu schicken und anschließend deren Ergebnisse in aggregierter Form über eine Weboberfläche anzusehen.
- Er ist sehr modular aufgebaut und unterstützt deswegen verschiedene Versionen von Linux und i2b2 sowie mehrere Datenbanksysteme.
- Viele Fehler können automatisch erkannt und behoben werden. Falls ein solches automatisches Beheben mal nicht möglich ist, so werden in der Regel hilfreiche Fehlermeldungen mit Schritten zur manuellen Behebung des Problems ausgegeben.

Zu Beachten ist bei der Verwendung des i2b2 Wizard noch, dass ein damit eingerichteter i2b2-Server nicht sicher ist und vor der Verwendung mit sensiblen Patientendaten weitere, manuelle Konfigurationsschritte erforderlich sind. Trotzdem stellt dieses Tool eine erhebliche Erleichterung bei der Verwendung von i2b2 dar, da andernfalls nahezu sämtliche der obigen Schritte und alles, was damit in Verbindung steht, manuell nacheinander ausgeführt werden müssen.

3 PaDaWaN

Der PaDaWaN [2], was eine Abkürzung für „Patient Data Warehouse Navigator“ ist, stellt ebenfalls ein Data Warehouse-System zur Verwaltung und vor allem zum Durchsuchen medizinischer Daten dar. Dieses System wird zur Zeit am Universitätsklinikum in Würzburg eingesetzt. Im Folgenden wird sowohl der Aufbau des Systems als auch die Funktionen genauer erläutert.

3.1 Aufbau

Zunächst werden nun die verschiedenen Bestandteile des PaDaWaN genauer beschrieben. Alle Erläuterungen hierzu basieren direkt auf dem PaDaWaN-Code[3] sowie der dazu zur Verfügung gestellten Dokumentation[4].

3.1.1 Datenbank

Genau wie bei der Standardimplementierung von i2b2 bildet den Kern des PaDaWaN eine Datenbank (genannt „PaDaWaN-Core“), in welcher sämtliche mit dem Data Warehouse in Verbindung stehende Daten gespeichert werden. Dies ist in der Regel eine MSSQL-Datenbank, kann aber auch eine MySQL-Datenbank sein. Die folgenden Erläuterungen beziehen sich auf einen „PaDaWaN-Core“ in der Version 1.1.

3.1.1.1 Katalog-Tabellen

Eine der wichtigsten Tabellen ist die **DWCatalog**-Tabelle, in welcher der komplette Katalog an möglichen Arten von Werten der medizinischen Fakten gespeichert ist. Sie setzt sich aus den folgenden Spalten zusammen:

- **AttrID** enthält die eindeutige interne Kennung eines Katalogeintrags. Diese Spalte wird in der Regel vom Datenbanksystem automatisch befüllt und stellt außerdem den Primary-Key der Tabelle dar.
- **ParentID** speichert die *AttrID* des Eintrags im Katalog, der in der Hierarchie aller Einträge direkt über dem aktuellen Eintrag steht. So besteht die Möglichkeit die Einträge über beliebig viele Hierarchieebenen hinweg einzuordnen und zu gruppieren.
- In der Spalte **Name** steht der eigentliche Name jedes Eintrags, der zum Beispiel in den graphischen Oberflächen angezeigt wird.
- Die Spalte **DataType** enthält den Datentyp, den zugehörige Werte haben. Hierfür gibt es die folgenden Möglichkeiten:
  - **SingleChoice** meint, dass der Wert nur aus einer vorgegeben Auswahl an festen Werten ausgewählt werden kann. Ein Beispiel hierfür wäre ein Eintrag für das Geschlecht eines Patienten, welcher dann als Wert nur entweder „männlich“ oder „weiblich“ haben kann.
  - Werte vom Typ **Text** können einen Text in beliebiger Form und Länge enthalten. Dies wird zum Beispiel benutzt, um komplette Arztbriefe abspeichern zu können.
  - Numerische Werte wie zum Beispiel Laborwerte können dem Datentyp **Number** zugeordnet werden.
  - Der Datentyp **Bool** wird für Katalogeinträge verwendet, deren Wert nur „Wahr“ oder „Falsch“ sein kann. Beispielhaft hierfür wäre ein Eintrag, der speichert, ob ein Patient bereits verstorben ist, oder nicht.

[3] Einsehbar mit entsprechendem Zugang unter https://gitlab2.informatik.uni-wuerzburg.de/misbased/

[4] Einzusehen nach entsprechender Freigabe unter https://oc.informatik.uni-wuerzburg.de

    - Katalogeinträgen mit dem Datentyp **Structure** sind in der Regel keine Werte direkt zugeordnet. Dieser Datentyp wird hauptsächlich für Einträge verwendet, die recht weit oben in der Kataloghierarchie stehen und nur zur Strukturierung und Gruppierung der Einträge auf den Ebenen darunter dienen.
    - **DateTime** ist der Datentyp, dessen Werte bestimmte Zeitpunkte enthalten. Ein Beispiel hierfür wäre der Zeitpunkt, zu dem ein Patient aus dem Krankenhaus entlassen wurde.
- **ExtID** ist eine Bezeichnung beziehungsweise eine Kennung für den Eintrag in dem Katalog, aus dem er in den PaDaWaN-Katalog übernommen wurde.
- **Project** speichert den Namen der Domäne oder des Projekts, dem sich der Eintrag zuordnen lässt. Das kann zum Beispiel „Diagnose“ sein. *Project* und *ExtID* zusammen können alternativ zur *AttrID* genutzt werden, um einen Eintrag eindeutig zu identifizieren.
- **OrderValue** ist ein numerischer Werte, der verwendet wird, um die Reihenfolge, in der die Einträge unter einem bestimmten Elterneintrag in der graphischen Oberfläche angezeigt werden, festlegen zu können.
- Die Spalte **UniqueName** enthält einen Namen des Eintrags, der in der Regel im kompletten Katalog eindeutig ist.
- Unter **Description** kann ein beliebiger Text zur weiteren Beschreibung eines Katalogeintrags gespeichert werden.
- Die **CreationTime** gibt den Zeitpunkt an, zu dem der Eintrag angelegt wurde.

Neben dieser Tabelle gibt es noch die folgenden drei Tabellen, die zusätzliche Informationen zu einzelnen Katalogeinträgen enthalten können und diesen mittels deren *AttrID* zugeordnet werden:

- **DWCatalogChoices:** In dieser Tabelle stehen die einzelnen Werte, die für Katalogeinträge mit dem Datentyp *SingleChoice* möglich sind. Da hier pro *AttrID* mehrere Einträge vorhanden sein können, haben diese jeweils noch eine eindeutige *ChoiceID*, die auch den Primary Key dieser Tabelle darstellt.
- **DWCatalogCount:** Hiermit kann für jeden einzelnen Eintrag gespeichert werden, wie viele zugehörige Patienten, Fälle und Werte insgesamt es im Data Warehouse gibt.
- **DWCatalogNumData:** Mit dieser Tabelle können für numerische Katalogeinträge Grenzwerte gespeichert werden, ab denen ein zugehöriger Wert als zu hoch oder zu niedrig gilt. Außerdem kann in dieser Tabelle die Einheit gespeichert werden, in der diese Werte gemessen werden.

3.1.1.2 Info-Tabelle

Die zweite zentrale Tabelle des PaDaWaN neben der *DWCatalog*-Tabelle ist die **DWInfo**-Tabelle. In ihr werden die einzelnen konkreten medizinischen Werte mithilfe der folgenden Spalten gespeichert:

- Die **InfoID** ist eine eindeutige Kennung jedes einzelnen Fakts, die den Primary Key der Tabelle darstellt. Sie ist nur zur internen Organisation der gespeicherten Informationen gedacht.

- **AttrID** enthält die Kennung des Katalogeintrags, auf den sich der gespeicherte Wert bezieht.
- Die **PID** ist die eindeutige Kennung des Patienten, für den der Fakt angelegt wurde.
- Mithilfe der **CaseID** können die Werte eines Patienten zusätzlich einzelnen Fällen zugeordnet werden. Diese Spalte enthält dann die eindeutige Kennung des Falls.
- Die Spalte **Ref** erlaubt dann eine noch feinere Einordnung eines Wertes, in dem sie in der Regel die eindeutige Kennung eines Dokuments innerhalb eines Falls enthält, aus welchem der Wert stammt.
- **MeasureTime** ist der Zeitpunkt, zu dem der Wert gemessen oder allgemein von einem Krankenhausmitarbeiter angelegt wurde.
- Zum Zeitpunkt, der in **ImportTime** gespeichert wird, wurde der Fakt in die Datenbank importiert.
- **Value** enthält den eigentlichen und vollständigen Wert des Fakts.
- **ValueShort** ist der Wert von *Value* auf dessen erste 100 Zeichen gekürzt.
- Mithilfe von **ValueDec** werden numerische Werte auch in numerischer Form in der Datenbank gespeichert.

3.1.1.3 Weitere Tabellen

Neben den obigen Tabellen gibt es noch einige weitere, die für zusätzliche Funktionen des PaDaWaN genutzt werden. Diese werden im Folgenden beschrieben:

- Es gibt verschiedene Möglichkeiten, um medizinische Daten in das PaDaWaN-System zu importieren. Hierfür gibt es unter anderem mehrere sogenannte Importer um unterschiedliche Arten von Daten einlesen zu können. Die einzelnen Importvorgänge werden in Verbindung mit ihrem aktuellen Status sowie einem eventuell dabei aufgetretenen Problem in der **DWImportLog**-Tabelle gespeichert. Während des Importierens verwenden die Importer außerdem die Tabellen **DWImportPIDs**, **DWImportCases** und **DWImportDocs**, um Metadaten zu einzelnen Patienten, Fällen sowie Dokumenten abfragen zu können. Beispiele hierfür sind das Geburtsjahr eines Patienten, die Art eines Falls sowie den Zeitpunkt der Erstellung eines Dokuments. Außerdem gibt es noch die Tabelle **DWRefID**, die während eines Importvorgangs das Abfragen eines bisher noch nicht verwendeten Wertes für die *Ref*-Spalte eines Fakts ermöglicht.
- Mit der Tabelle **DWQuery** können beliebige Suchanfragen in Verbindung mit einem Namen, dem Zeitpunkt des ersten Speicherns und dem Zeitpunkt der letzten Veränderung der Anfrage zum späteren Abrufen gespeichert werden. Die Tabelle **DWQueryLog** enthält hingegen sämtliche ausgeführten Anfragen zusammen mit dem Zeitpunkt der Ausführung, dem Nutzer, der die Anfrage gestellt hat und dem Namen des Rechners, von dem die Anfrage kam.
- In der Tabelle **DWSystemParams** können für beliebige Parameter beliebige Werte abgelegt werden. Das wird zum Beispiel genutzt, um das Datum der letzten Indexierung der Daten mit einer der weiter unten noch beschriebenen Suchengines zu speichern und abzufragen.

– Der PaDaWan ermöglicht auch eine Nutzerverwaltung mithilfe der Tabelle **DWUsers**. Hier wird neben dem Benutzernamen und dem Passwort auch der vollständigen Name und die E-Mail-Adresse gespeichert. In der Tabelle **DWUserSettings** ist dann für jeden Nutzer hinterlegt, ob er zu den Katalogeinträgen in der graphischen Oberfläche die Zahl an passenden Patienten, Fällen oder die aller zugehöriger Fakten angezeigt bekommen möchte. Auch ist hier gespeichert, ab welcher Mindestanzahl der Nutzer einen Eintrag überhaupt angezeigt bekommen möchte. Die Nutzer werden mithilfe der Tabelle **DWUserInGroup** einer Gruppe oder auch mehreren Gruppen zugeordnet, die wiederum in der **DWGroup**-Tabelle gespeichert sind. Jede Gruppe hat darin einen Namen sowie die folgenden Parameter:
  - **kAnonymity:** Wenn dieser Wert zum Beispiel auf 10 gesetzt ist, so werden die Ergebnisse aller Teilanfragen von ausgeführten Verteilungssuchen (werden weiter unten noch beschrieben), die weniger als 10 Ergebnisse liefern, zensiert. Bei einem Wert von 0 für diesen Parameter werden immer alle Ergebnisse angezeigt.
  - **case_query:** Dies ist ein boolescher Wert, der angibt, ob ein Nutzer in dieser Gruppe auch dazu berechtigt ist, sich die konkreten Werte der von einer Anfrage gefundenen Fakten anzeigen zu lassen, oder ob er nur aggregierte Zahlen als Ergebnis von Verteilungssuchen sehen darf.
  - **admin:** Dieser boolesche Wert gibt an, ob ein Nutzer der Gruppe Zugang zur Administrationsoberfläche und den damit verbunden Möglichkeiten zur Nutzerverwaltung hat oder nicht.

  Für die Gruppen gibt es außerdem noch die Tabellen **DWGroupCasePermission** und **DWGroupCatalogPermission**, die entweder in Form einer White- oder Blacklist alle Fallkennungen sowie Katalogeinträge enthalten, die die Nutzer einer Gruppe in der graphischen Oberfläche überhaupt angezeigt beziehungsweise nicht angezeigt bekommen.

3.1.2 MXQL

Während - wie oben bereits beschrieben - die Suchen nach Fakten nur einen Teil der möglichen Anfragen an die CRC-Komponente ausmachen, gibt es zu diesem Zweck im PaDaWaN eine eigene (ebenfalls XML-basierte) Anfragesprache namens MXQL („Medical XML Query Language“). MXQL legt die Möglichkeiten der Suchanfragen fest, die sich aus den folgenden Elementen (bezogen auf MXQL-Version 0.7) zusammensetzen können:

– Jedes Element der Suche kann das Attribute **active** besitzen. Damit besteht die Möglichkeit, ein Element samt aller untergeordneten Elemente zu deaktivieren und somit im Rahmen der Suche nicht zu berücksichtigen.
– Mit **comment** kann zu jedem Element ein beliebiger Kommentar zu dessen Erläuterung hinzugefügt werden.
– **optional** ermöglicht es ein Element, inklusive aller untergeordneter Elemente als optional zu markieren. Ein optionales Element muss nicht bei zum Beispiel einem Patienten gefunden werden, damit dieser als Treffer für die Suche zählt. Kommt es jedoch bei ihm vor, so wird es mit zurückgegeben.

- Alle Bestandteile der Suche können die folgenden **Sub-Elemente** enthalten:
  - **TempOpAbs:** Dieses Element schränkt ein anderes auf einen bestimmten Zeitraum ein. Hierbei kann entweder eine Untergrenze, eine Obergrenze oder beides angegeben werden. So werden zum Beispiel nur Patienten gefunden, bei denen ein gewisser Wert vor einem bestimmten Zeitpunkt auftrat. Es können auch mehrere dieser Elemente einem anderen hinzugefügt werden, wodurch sie dann verodert werden.
  - **TempOpRel:** Hiermit kann der Zeitpunkt eines Attributs relativ zu einem anderen eingeschränkt werden. Dazu wird die gesuchte Zeitspanne definiert, was dann zum Beispiel ermöglicht nur Fälle zu finden, in denen ein Attribut höchstens ein Jahr nach einem anderen auftritt. Auch hier werden mehrere dieser Elemente verodert.
- Das **Query**-Element ist das Element der Suche, das immer auf der höchsten Ebene der XML-Anfrage vorkommt. Es bietet die folgenden Parameter:
  - **pid** wird auf „true“ gesetzt, wenn für jedes gefundene Ergebnis auch die Kennung des zugehörigen Patienten zurückgegeben werden soll.
  - Wenn **onlyCount** auf „true“ steht, dann werden anstatt der gefundenen Werte nur die Anzahl an insgesamt gefundenen Dokumenten zurückgegeben.
  - Der Parameter **limit** legt fest, wie viele Ergebnisse zurückgegeben werden sollen. Mit einer *0* hierbei erhält man sämtliche Ergebnisse zurück.
  - Mit **distinct** kann angegeben werden, dass pro Patientenkennung nur ein Ergebnis zurückgegeben werden soll. Bei mehreren Treffern pro Patient wird dann nur der mit dem kleinsten Wert ausgegeben.
  - **version** legt die MXQL-Versionsnummer fest, nach deren Definition die aktuelle Anfrage zusammengebaut ist. Aktuell ist dies die Versionsnummer *0.7*.
- Mit einem oder mehreren **Attribute**-Elementen wird festgelegt, was eigentlich von der Suche gefunden werden soll. Hierzu stehen bei diesem Element die folgenden Parameter zur Verfügung:
  - **extID** und **domain** beziehen sich auf die Spalten *ExtID* und *Project* der Katalogtabelle und legen den zu suchenden Katalogeintrag fest. Sie müssen immer beide angegeben werden.
  - **elementID** erlaubt die Benennung und damit auch die Referenzierung eines Attributs für eine relative Suche wie zum Beispiel mit dem oben beschriebenen *TempOpRel*.
  - Der Parameter **desiredContent** gibt an, was für einen Wert der gefundene Fakt haben muss, um als Suchtreffer zu zählen. Wenn der Katalogeintrag numerisch oder ein Zeitpunkt ist, so kann hier auch ein Wertebereich angegeben werden, in dem die Treffer liegen sollen. Bei einem Text-Datentyp können beliebig viele Wörter angegeben werden, die dann in beliebiger Form im Text des Fakts enthalten sein müssen. Hierbei können einzelne Wörter auch den Wildcard-Operator * enthalten, der dann für eine beliebige Zeichenkette steht. Bei mehreren Wörtern ist es außerdem möglich anzugeben, dass diese in der angegebenen Reihenfolge und/oder in einem gewissen Abstand zueinander vorhanden sein müssen.

- Der **contentOperator** gibt an, wie der *desiredContent* verarbeitet werden soll. Er gibt zum Beispiel an, dass der Wert gefundener Fakten größer, kleiner oder genau gleich sein muss. Auch kann hiermit angegeben werden, dass der angegebene *desiredContent* nicht in den Werten enthalten sein darf oder dass dessen Wert nur in einem negierten oder nicht negierten Kontext vorkommen darf. Das erlaubt beispielsweise die Suche nach „Fieber“ ohne das Fakten mit Werten wie „kein Fieber“ gefunden werden.
- Mit dem **reductionOp** wird angegeben, was bei mehreren Werten pro zum Beispiel gefundenem Patient geschehen soll. Hier kann entweder nur der minimale, maximale, früheste oder späteste Wert ausgegeben werden. Wenn kein *reductionOp* angegeben wird, so werden in Abhängigkeit von dem Parameter **multipleRows** die Werte entweder mit Kommata getrennt oder es wird für jeden Wert eine separate Zeile im Ergebnis erzeugt.
- **valueInFile** wird auf „true“ gesetzt, wenn der für das aktuelle Attribut gefundene Text nicht direkt im Ergebnis dargestellt sondern stattdessen in eine Datei geschrieben soll und im Ergebnis nur eine Verknüpfung auf diese Datei zu sehen sein soll.
- Mit **infoDate**, **caseID** und **docID** lässt sich angeben, ob für jeden gefundenen Fakt auch dessen Messzeitpunkt und seine Fall- beziehungsweise Dokumenten-Kennung zurückgegeben werden soll.
- **displayValue** legt fest, ob die Werte der zum aktuellen Attribut passenden Fakten überhaupt ausgegeben werden sollen. Alternativ lässt sich gerade für längere Texte **onlyDisplayExistence** verwenden, um anstelle des kompletten Werts nur ein „x“ zurückzugeben.
- Mit **searchSubEntries** kann angegeben werden, ob ein Fakt nur dann als Treffer zählt, wenn er sich direkt auf den angegebenen Katalogeintrag bezieht oder ob er auch dann gefunden werden soll, wenn er sich auf einen in der Kataloghierarchie unter dem Gesuchten liegenden Eintrag bezieht.
- Die Parameter **minCount** und **maxCount** geben an, wie oft ein Attribut mindestens beziehungsweise höchstens pro Treffer vorkommen muss beziehungsweise darf, damit dieser Treffer zurückgegeben wird. Der Parameter **countType** legt dabei fest, ob sich diese Zählung auf die Anzahl pro Patient, Fall, Dokument oder Jahr bezieht.
- Bei einer Suche innerhalb eines Textes wird in der Regel der oder die gefundenen Begriffe im Text markiert und ein Teil des Textes vor und nach den einzelnen Treffern zurückgegeben. Der Parameter **extractionMode** ermöglicht alternativ entweder die Rückgabe der nächsten Zahl nach einem Treffer innerhalb des Textes oder die Rückgabe des Textes zwischen zwei Treffern.
- Jedes *Attribute*-Element kann untergeordnet noch beliebig viele **ValueCompare**-Elemente besitzen. Hiermit ist es möglich den Wert des aktuellen Attributs in Relation zu einem anderen einzuschränken. Zum Beispiel werden so nur Werte gefunden, die größer als die Werte des referenzierten Attributs sind.

– Die Elemente **And** und **Or** erlauben eine logische Gruppierung der Elemente der Suche. Diese können auch beliebig kombiniert und verschachtelt werden und finden dann nur Treffer, bei denen entweder alle oder mindestens ein untergeordnetes Element zutrifft. Wenn kein *And* oder *Or* in einer Anfrage vorkommt, dann werden standardmäßig alle Anfragenelemente mittels *And* verbunden. Außerdem gibt es noch ein **Not**-Element, welches nur genau ein untergeordnetes Element haben darf und bewirkt, dass dieses in negierter Form zutreffen muss.
– Mit **IDFilter**-Elementen können die Treffer der untergeordneten Elemente auf fixe Patienten-, Fall- beziehungsweise Dokumentenkennungen beschränkt werden. Zusätzlich können die Kennungen optional auf einen bestimmten Messzeitpunkt eingegrenzt werden. Alternativ lassen sich hiermit die Treffer auch auf einzelne fixe Jahre beschränken. Auch dieses Element besitzt einen *distinct*-Parameter, der die Ergebnisse auf einen Treffer pro Patient, Fall, Dokument oder Jahr einschränkt. Außerdem legt dieses Element generell fest, ob von der Suche passende Patienten, Fälle, Dokumente oder Jahre zurückgegeben werden sollen. Ohne ein solches Element sind das standardmäßig immer Patienten.
– **SubQuery** erlaubt die Einbindung einer kompletten MXQL-Anfrage als untergeordnetes Element einer neuen Suche.
– Mit **DistributionColumn**, **DistributionRow** und **DistributionFilter** können „normale“ Anfragen zu sogenannten „Verteilungssuchen“ gemacht werden. Hierbei wird die Anfrage mehrfach mit unterschiedlichen Einschränkungen gestellt um so zum Beispiel die Verteilung eines Attributs über mehrere Jahre hinweg abzufragen. Dabei wird dann die Anfrage für jedes betrachtete Jahr einmal gestellt und anschließend die Anzahl an Treffer für jedes Jahr separat zurückgegeben.
– Mit dem **SortOperator** kann angegeben werden, wonach (Wert, Messzeitpunkt, Patienten- oder Fallkennung) und wie (aufsteigend oder absteigend) die Treffer der einzelnen Attribute sortiert werden sollen.

3.1.3 Web- und RCP-Oberfläche

Der PaDaWaN bietet zwei unterschiedliche graphische Benutzeroberflächen zum Stellen von Anfragen. Sie sind in den Abbildungen 8 und 9 zu sehen. Während eine der Oberflächen webbasiert ist, stellt die andere eine native Anwendung basierend auf der „Eclipse Rich Client Platfrom“[5] (RCP) dar. Von der Weboberfläche wird im Folgenden die Version 0.0.1 und von der nativen Oberfläche die Version 0.5 genauer beschrieben.
Beide bieten im linken Bereich die Möglichkeit, die Kataloghierarchie anzusehen und durchzugehen sowie zu durchsuchen und aus einzelnen Einträgen eine neue Anfrage zusammenzubauen. Auch lässt sich bei beiden einstellen, welche Anzahl neben den Einträgen angezeigt werden soll, wobei man zwischen der Anzahl passender Patienten und Fälle sowie der Anzahl aller Fakten zu jeweils einem Eintrag wählen kann. Außerdem lässt sich konfigurieren, dass Einträge mit einer

[5] https://wiki.eclipse.org/Rich_Client_Platform

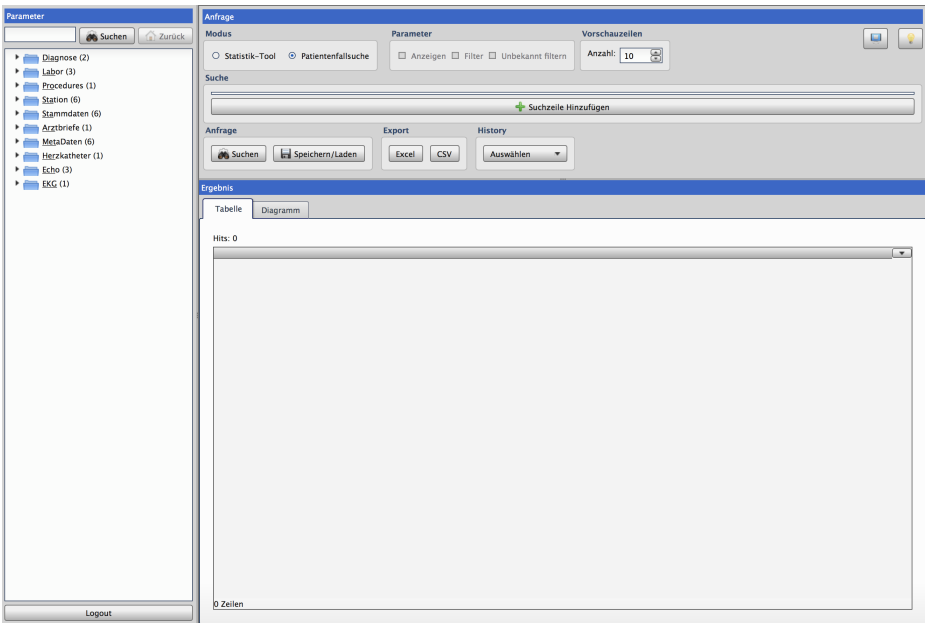

Abb. 8. Weboberfläche des PaDaWaN (eigener Screenshot)

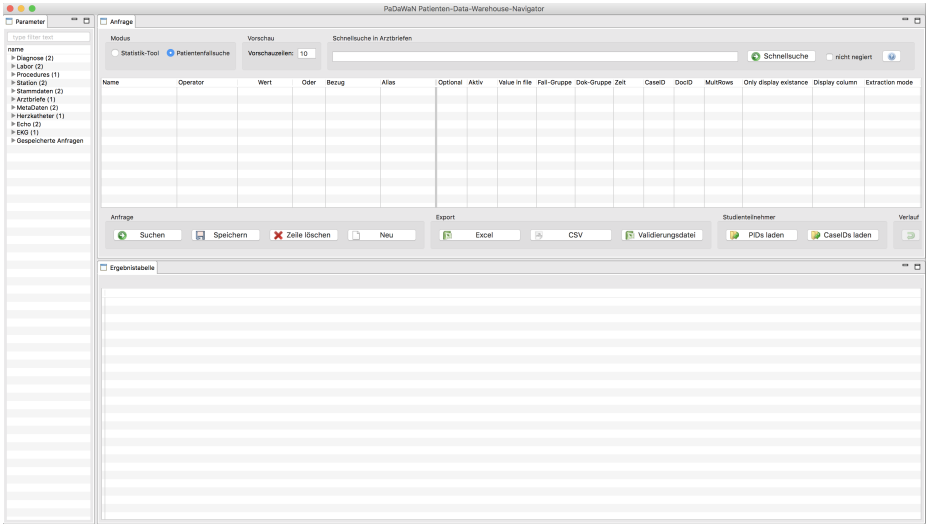

Abb. 9. RCP-Oberfläche des PaDaWaN (eigener Screenshot)

Anzahl unter einem bestimmten Grenzwert gar nicht in der Oberfläche angezeigt werden sollen.
In der RCP-Oberfläche werden an dieser Stelle auch gespeicherte Anfragen angezeigt, die von dort aus auch in eine neue Anfrage übernommen werden können. Neue Anfragen können in beiden Oberflächen über einen entsprechenden Button gespeichert werden. Die Weboberfläche nutzt die Ansicht zum Speichern von Anfragen auch zum Laden bereits gespeicherter Anfragen sowie zu deren Einbindung in eine neue Suche.
Im oberen Bereich der Oberflächen werden die Anfragen erstellt und konfiguriert. Dazu lassen sich über die RCP-Oberfläche direkt ein Großteil der MXQL-Parameter für jedes Attribut der Anfrage mittels Checkboxen, Drop-Down-Menüs und Textfeldern konfigurieren. Auch gibt es hier eine sogenannte Schnellsuche, die eine Suche nach verschiedenen Begriffen in Arztbriefen automatisch in einzelne MXQL-Attribute zerlegt. Die genaue Syntax dieser Suche lässt sich über eine Verknüpfung auf die zugehörige Webdokumentation einsehen. Die RCP-Oberfläche erlaubt außerdem das Einlesen von Dateien mit festen Patienten- und Fallkennungen, um die Suchergebnisse auf diese einzuschränken. In der Weboberfläche werden alle Attribute in einer eigenen Suchsprache genannt „Quick-Search“ definiert. Auch für diese Sprache ist eine Webdokumentation aufrufbar und sie erlaubt ebenfalls die Verwendung eines Großteils der Möglichkeiten von MXQL.
Neben der Suche nach einzelnen Werten medizinischer Fakten (genannt „Patientenfallsuche“) ist auch die Verteilungssuche (in den Oberflächen „Statistik-Tool“ genannt) in beiden Oberflächen verfügbar. Außerdem lässt sich jeweils die Anzahl an zurückzugebenden Suchtreffern festlegen.
Im unteren Teil beider Oberflächen werden die Suchergebnisse in tabellarischer Form dargestellt. Die Ergebnisse lassen sich dort jeweils auch entweder als Excel- oder als CSV-Datei exportieren. Die Ergebnisse der Verteilungssuche können zusätzlich auch als Diagramm angezeigt werden.
Als weitere Funktion speichern beide Oberflächen einen Verlauf der letzten Anfragen und Ergebnisse und ermöglichen deren erneutes Anzeigen.
Über die Einstellungen erlaubt die RCP-Oberfläche zusätzlich noch das Wechseln der im Hintergrund genutzten Suchengine sowie das Exportieren des kompletten Katalogs im CSV-Format.
Die Weboberfläche bietet eine separate Ansicht zum Verwalten von Nutzern und Nutzergruppen sowie deren Rechte und Zuordnungen zueinander.

3.2 Verwendete Engines

Für Suchen mit dem PaDaWaN können verschiedene sogenannte Engines genutzt werden, die im Folgenden genauer vorgestellt werden.

- Als einfachste Möglichkeit erlaubt der PaDaWaN, dass MXQL-Anfragen in entsprechende **SQL**-Statements umgewandelt werden können, um so die Datenbank direkt zu durchsuchen. Mit dieser Engine werden die meisten MXQL-Funktionen mit Ausnahme des Hervorhebens von Treffern und der

Suche nach mehreren Begriffen innerhalb eines bestimmten Umkreises zueinander unterstützt. Auch die relativen Suchen sind hiermit nicht möglich. Da Datenbankabfragen jedoch gerade bei größeren Datenmengen oft sehr langsam sind, werden für Suchen in der Regel die folgenden Engines verwendet. Anzumerken ist hierbei noch, dass die als „nicht unterstützt/möglich“ bezeichneten Funktionen nur mit der aktuellen PaDaWaN-SQL-Engine nicht genutzt werden können. Auf einer SQL-Datenbank sind solche Anfragen grundsätzlich schon möglich.

- **Solr** [38] ist eine eigenständige Open Source Such-Lösung basierend auf Apache Lucene [39]. Eine der größten Stärken dieses Systems ist neben der hohen Geschwindigkeit, mit der Anfragen verarbeitet werden, die Möglichkeiten zur Volltextsuche. Im PaDaWaN wird diese vor allem dazu genutzt, um bei Suchen nach einzelnen Worten in längeren Texten nicht den kompletten Text sondern nur die Stellen mit den gefundenen Suchbegriffen zurückzugeben. Eine weitere Möglichkeit von Solr ist die Suche nach mehreren Wörtern mit einem gewissen Maximalabstand zueinander. Die oben beschriebenen relativen Suchen von MXQL sind hiermit jedoch nicht direkt möglich und müssen stattdessen in einer separaten Nachbearbeitung der Ergebnisse gemacht werden. Solr ist die zurzeit am häufigsten im PaDaWaN eingesetzte Suchengine.
- Auch **Elasticsearch** [40] basiert auf Apache Lucene. Neben den Möglichkeiten von Solr können einzelne Dokumente im Index dieser Engine in einer hierarchischen Struktur mit Beziehungen zueinander angelegt werden, wobei immer ein Dokument auf einer oberen Ebene beliebig viele untergeordnete Elemente haben kann. Dies erlaubt eine Strukturierung des Indexes mit einer separaten Fall- und Patientenebene, um dann auch Patienten zu finden, wenn die Attribute der Anfrage fallübergreifend vorkommen. Um dies mit Solr zu erreichen müssen alle Fakten sowohl für die Patienten als auch für die Fälle separat gespeichert werden. Auch mit Elasticsearch sind relative Suchen nur mittels einer separaten Nachbearbeitung möglich. Durch die hierarchische Indexstruktur ist Elasticsearch allerdings etwas langsamer als Solr.
- **Neo4j** [41] ist ein graph-basierte Suchlösung bei der zwischen einzelnen Dokumenten im Index (genannt Knoten) beliebige Beziehungen gespeichert und bei Anfragen berücksichtigt werden können. Somit erlaubt sie das direkte Ausführen von relativen Suchen und verarbeitet diese in der Regel deutlich schneller als die Datenbank bei der SQL-Engine. Auch ist dadurch eine ähnliche hierarchische Indexstrukturierung wie bei Elasticsearch möglich. Allerdings ist auch hiermit kein Hervorheben von Suchtreffern und keine Umkreissuche in Texten möglich. Neo4j ist bei nicht-relativen Suchen außerdem langsamer als Elasticsearch und Solr und wird deswegen vor allem für relative Anfragen verwendet.

4 Vergleich zwischen i2b2 und PaDaWaN

In diesem Kapitel werden die beiden System im Bezug auf unterschiedliche Aspekte miteinander verglichen. Sofern nicht anders angegeben basieren die Ausführungen hierzu auf den obigen Beschreibungen beider Systeme und den dort

angegeben Quellen. Die Erläuterungen beziehen sie alle auf i2b2 in Version 1.7.07 sowie einen „PaDaWaN-Core“ in Version 1.1.

4.1 Datenstrukturen

Den Kern beider Systeme bildet standardmäßig eine Datenbank. Sowohl i2b2 als auch der PaDaWaN basieren auf einerseits einem Katalog medizinischer Konzepte und andererseits einer Menge medizinischer Fakten. Im Folgenden werden die Unterschiede und Gemeinsamkeiten der Speicherung dieser beiden Aspekte genauer betrachtet.

4.1.1 Katalog medizinischer Konzepte

Der Katalog legt jeweils fest, was für Arten von medizinischen Fakten überhaupt in den System abgelegt werden können. In beiden Systemen ist er hierarchisch aufgebaut und erlaubt so eine Strukturierung und Gruppierung der einzelnen Konzepte, aus welchen sich dann auch die Suchanfragen zusammensetzen.
Die hierarchische Struktur wird beim PaDaWaN durch ein Angeben der numerischen Kennung (*AttrID*) des übergeordneten Eintrages (*ParentID*) erzeugt. Da diese Kennung in der Regel von der Datenbank selbst vergeben wird, steht sie allerdings in keinem direkten Bezug zu dem Eintrag und kann sich bei einer Neuanlegung des Katalogs auch ändern. In i2b2 ist jeder Eintrag durch einen eindeutigen Pfad spezifiziert. Mithilfe dieses Pfades entsteht hier die Hierarchie. Das hat den Vorteil, dass ein Katalogeintrag immer dieselbe Kennung behält. Ein Nachteil der Verwendung des Pfades ist aber, dass Verschiebungen von Einträgen innerhalb der Hierarchie schwieriger sind als im PaDaWaN, da dann der Pfad aller untergeordneten Einträge ebenfalls geändert werden muss. Im PaDaWaN ist eine solche Verschiebung mit dem Ändern einer einzelnen *ParentID* möglich.
Über *ExtID* und *Project* ist jeder PaDaWaN-Eintrag allerdings immer eindeutig identifizierbar. Eine ähnliche Funktion erfüllt der *C_BASECODE* in i2b2, der jedoch nicht bei allen Einträgen benutzt wird, da nur in i2b2 Konzepte angelegt werden können, die nicht als Teil einer Suche benutzt werden dürfen und somit auch keinen *C_BASECODE* benötigen.
Der *UniqueName* des PaDaWaN erfüllt eine ähnliche Funktion wie *C_SYMBOL* in i2b2 (was dem letzten Teil des oben erwähnten Pfades entspricht), da beide eine weitere eindeutige Bezeichnung eines Eintrages darstellen. In beiden Systemen kann es vorkommen, dass einzelne Einträge denselben Namen haben. Sowohl der *UniqueName* des PaDaWaN, als auch das *C_SYMBOL* von i2b2 werden dann um Zahlen zur Unterscheidung dieser Einträge ergänzt.
Analog in beiden Systemen funktionieren *C_COMMENT* und *Description*, *C_NAME* und *Name*, *C_TOTALNUM* und die Anzahl passender Patienten sowie *IMPORT_DATE* und *CreationTime*.
Im Hinblick auf die unterstützten Datentypen bietet PaDaWaN mit *DateTime*, *Bool*, *Structure* und *isA* einige, die in der Form nicht direkt von i2b2 unterstützt werden. Da die zugehörigen Werte aber nur in Textform gespeichert werden,

beziehungsweise es nur darum geht, das ein Fakt zu diesem Attribut überhaupt vorkommt, lässt sich das analog auch mit dem *String*-Datentyp von i2b2 umsetzen. Dieser Datentyp entspricht im PaDaWan dem Typ *Text*. Bei i2b2 kann hierbei noch die maximal zulässige Textlänge angegeben werden. Der PaDaWaN-Datentyp *SingleChoice* entspricht dem *Enum*-Datentyp von i2b2. Im Hinblick auf numerische Daten bietet i2b2 noch die Einschränkung auf positive sowie auf ganzzahlige Werte, die im PaDaWaN nicht explizit mit angegeben werden kann. Auch in beiden Systemen lassen sich für Einträge zu numerischen Fakten Extremwerte angeben, ab denen zum Beispiel ein Laborwert zu niedrig ist. Im PaDaWaN kann hierzu jeweils eine obere und eine untere Grenze angegeben werden. i2b2 ermöglicht die Aufteilung dieser beiden Grenzen in drei separate Grenzen, um so zum Beispiel noch festzulegen, ab wann ein Wert nur etwas erhöht und ab wann er so sehr erhöht ist, dass diese Erhöhung für den Patienten ein Risiko darstellt. Zu diesen numerischen Einträgen kann im PaDaWaN eine und i2b2 mehrere Einheiten (inklusive des Umrechnungsfaktors dazwischen) angegeben werden, in denen gemessen wird. Die verwendete Einheit wird in i2b2 dann auch als Teil jedes Fakts zu einem Konzept mit mehreren möglichen Einheiten gespeichert.
Die folgenden Funktionen werden ausschließlich vom PaDaWaN unterstützt:

- Der *OrderValue* erlaubt ein Festlegen der Anzeigereihenfolge der Einträge in einer graphischen Benutzeroberfläche.
- Neben der Anzahl an passenden Patienten können für jeden Eintrag noch die Anzahl passender Fälle sowie die Anzahl aller passenden Fakten gespeichert werden.

Manche Funktionen werden auch nur von i2b2 unterstützt:

- Einzelne Katalogeinträge können beliebig viele Synonyme haben. Jedes der Synonyme kann in einer Suche nach Fakten verwendet werden und liefert dieselben Ergebnisse.
- Es lässt sich direkt angeben, ob sich ein Attribut auf unterster Hierarchieebene befindet, sodass dies in der Benutzeroberfläche anders dargestellt werden kann. Auch besteht die Möglichkeit, Konzepte als editierbar zu markieren, was dann die Möglichkeit zum Anpassen und Löschen über die graphische Oberfläche aktiviert.
- i2b2 verfügt noch über einige zusätzliche Spalten für verschiedene Metadaten, die aus Nutzersicht jedoch keine Rolle spielen und hauptsächlich für die interne Verarbeitung der Einträge genutzt werden.
- Einzelne Katalogeinträge können als Modfier für andere Einträge angelegt werden, um diese mit zusätzlichen Metadaten zu ergänzen. Ein Beispiel für einen solchen Modifier wird weiter oben bei der Erläuterung der Datenstruktur der i2b2-ONT-Komponente beschrieben.

Zusammenfassend fällt auf, dass sich ein PaDaWaN-Katalog größtenteils problemlos in i2b2 übernehmen lässt. Nur einige der Datentypen und gespeicherten Anzahlen an passenden Elementen lassen sich nicht direkt übernehmen. Während sich einerseits die fehlenden Datentypen recht einfach in i2b2 verwenden

lassen, da die zugehörigen Werte im PaDaWaN in der Regel immer als String gespeichert werden und somit dies als i2b2-Datentyp genutzt werden kann, gehen anderseits die zusätzlichen Anzahlen und die damit verbunden Funktionen der Benutzeroberflächen in i2b2 komplett verloren.
In die andere Richtung verliert man hauptsächlich die zusätzlichen Möglichkeiten für Metadaten von numerischen Werten (Extremwerte, zusätzliche Einheiten mit Umrechnungsfaktor, Einschränkung auf ganze und/oder positive Zahlen), die Beschränkung der Länge von Texten und die Möglichkeit Synonyme zu verwenden. Modifier sind nur eine spezielle Art von Katalogeinträgen und lassen sich auch problemlos als solche modellieren. Die restlichen Metadaten der i2b2-Konzepte sind teilweise redundant und lassen sich aus anderen Feldern wieder generieren. Die verbleibenden Felder werden genutzt um die *CONCEPT_DIMENSION*-Tabelle der CRC-Komponente bei einer Suchanfrage nach den passenden *C_BASECODE*-Werten zu durchsuchen. In der Standardimplementierung haben diese Felder in der Regel die gleichen Werte und müssen somit in den meisten Fällen nicht separat gespeichert werden.

4.1.2 Medizinische Fakten

Ein Hauptanwendungszweck beider Systeme ist die Auswertung medizinischer Fakten (im PaDaWaN „Informationen“genannt).
Viele der in den Fakten gespeicherten Kennungen kommen in beiden Systemen vor. So entsprechen sich *PATIENT_NUM* und *PID*, *ENCOUNTER_NUM* und *CaseID* sowie *CONCEPT_CD* und *AttrID*. Und auch wenn sie für etwas unterschiedliche Zwecke gedacht sind, so erfüllen auch *INSTANCE_NUM* und *Ref* eine ähnliche Funktion.
Die eigentlichen Werte werden ebenfalls in ähnlicher Form gespeichert. Dabei entsprechen sich *START_DATE* und *MeasureTime*, *NVAL_NUM* und *ValueDec* sowie für die meisten Fakten *TVAL_CHAR* beziehungsweise *OBSERVATION_BLOB* in i2b2 und *Value* im PaDaWaN.
Auch analog in beiden Systemen funktionieren *IMPORT_DATE* und *ImportTime*.
Einige Elemente der PaDaWaN-Informationen können allerdings nicht direkt in i2b2 übernommen werden:

- Die *InfoID* dient als Primary Key der Informationstabelle und zur PaDaWaN-internen Verwaltung der Informationen. In i2b2 gibt es keinen separaten Primary Key. Diese Funktion übernimmt hier eine Kombination aus mehreren der gespeicherten Kennungen. Deswegen kann die InfoID zwar nicht übernommen werden, es gehen aber damit auch keine Informationen verloren.
- Der *ValueShort* ist lediglich eine gekürzte Version des *Value*, falls gerade bei längeren Texten nicht der komplette Text zurückgegeben werden soll. Somit verliert man auch hier keine Daten. Da der *TVAL_CHAR* bei längeren Texten in i2b2 nur angibt, ob diese verschlüsselt sind (was im PaDaWaN nicht möglich ist), könnte auch dieser beim Übertragen von PaDaWaN-Datein wie ein *ValueShort* in i2b2 verwendet werden.

Während sich so sämtliche Daten der PaDaWaN-Informationen in i2b2 übernehmen lassen, ist dies in die andere Richtung nur teilweise möglich, da die folgenden Elemente im PaDaWaN fehlen:

- Einige Metadaten wie *UPDATE_DATE*, *DOWNLOAD_DATE*, *SOURCESYSTEM_CD* und *UPLOAD_ID* sind nur zur i2b2-internen Verwaltung der Fakten gedacht und können somit weggelassen werden.
- *PROVIDER_ID*, *LOCATION_CD*, *END_DATE* und *CONFIDENCE_NUM* sind alles zusätzliche Daten mit Bezug auf jeweils einen einzelnen Fakt. Diese Informationen gingen im PaDaWaN alle verloren. Das *END_DATE* kann auch in zeitlich relativen Suchen in i2b2 verwendet werden (wird weiter unten noch beschrieben), was folglich im PaDaWaN ebenfalls nicht möglich ist.
- Wie beim Vergleich der Kataloge schon beschrieben, können Modifier auch mit separaten „normale“ Konzepten umgesetzt werden, was die *MODIFIER_CD* überflüssig macht.
- Die Werte von *VALTYPE_CD* und *VALUEFLAG_CD* können auch mithilfe des zugehörigen Katalogeintrages ermittelt und somit ebenfalls weggelassen werden.
- *UNITS_CD* ist in i2b2 nötig, da es dort pro Katalogeintrag mehrere Einheiten für zugehörige Werte geben kann. Im PaDaWaN ist dies immer nur eine Einheit, weswegen diese Möglichkeit dort fehlt. Allerdings kann i2b2 auch Umrechnungsfaktoren zwischen den Einheiten speichern, wodurch Werte in anderen Einheiten vor dem Speichern im PaDaWaN umgerechnet werden können.
- Auch die *QUANTITY_NUM* lässt sich in den Wert der Information mit einrechnen.

i2b2 speichert in weiteren Tabellen noch zusätzliche Informationen zu den Patienten und Fällen im System. Die Werte die hierbei in den *DIMENSION*-Tabellen stehen, sind nur Fakten, die zusätzlich dort gespeichert wurden, um direkt grundlegende Informationen zu gefundenen Patienten beziehungsweise Fällen ausgeben zu können. All diese Daten sind auch noch als „normale“ Fakten gespeichert und können somit auch nach einer Übernahme der Daten in den PaDaWaN abgefragt werden und ließen sich selbst bei einem Rücktransfer der Daten in i2b2 wiederherstellen.
Mit den *MAPPING*-Tabellen ist es möglich den von i2b2 generierten Kennungen für Fälle und Patienten Kennungen aus anderen Systemen zuzuordnen. Dies ließe sich im PaDaWaN durch entsprechende Katalogeinträge für diese externen IDs übernehmen.
Texte können in i2b2 auch in verschlüsselter Form gespeichert und in der Oberfläche je nach Rechten des angemeldeten Nutzers wieder entschlüsselt dargestellt werden. Diese Texte können zwar auch im PaDaWaN abgelegt werden, wo es dann allerdings zurzeit keine eingebaute Möglichkeit zur Entschlüsselung gibt.
Zusammenfassend lassen sich PaDaWaN-Daten problemlos in ein i2b2-System übertragen, während dies in die andere Richtung mit einigen Umrechnungen

sowie mit dem Verlust mancher Daten verbunden ist. Zumindest auf die Suchfunktion nach Fakten hätte dieser Informationsverlust aber - mit Ausnahme des *END_DATE* - keine Auswirkungen.

4.2 Importmöglichkeiten

Sowohl bei i2b2 als auch beim PaDaWaN gibt es verschiedene Möglichkeiten neue Daten in die Systeme zu importieren. Da beide in ihrer Standardimplementierung auf eine Datenbank im Hintergrund setzen, lassen sich Daten jeweils mit eigenen Klassen oder Skripten sowie über die jeweiligen Administrationsoberflächen der Datenbanken direkt in diese importieren. Da hierbei aber die Daten entweder schon in genau zu den Systemen passender Form vorliegen oder sonst in diese manuell umgewandelt werden müssen, bieten beide Systeme verschiedene Möglichkeiten, um den Importvorgang zu vereinfachen.
Im PaDaWaN sind das die folgenden zwei:

- Mit sogenannten *konfigurierten Importern* können Daten im CSV- und XML-Format eingelesen werden. Dazu müssen diese Daten in einer einfachen und bestimmten Formatvorschriften folgenden Struktur vorliegen, in welcher die zu importierenden Informationen direkt enthalten sind. In diesem Fall muss der Importer lediglich konfiguriert werden und ist anschließend direkt verwendbar.
- Für Daten in beliebiger anderer und vor allem komplexerer Form können eigene Importer entwickelt werden, für die es bereits verschiedene abstrakte Klassen gibt, von welchen dann jeweils nur noch einige Methoden implementiert werden müssen.
- Eine eigene REST-Schnittstelle zum Importieren von Daten wird für den PaDaWaN zur Zeit noch entwickelt.

Es existieren im PaDaWaN bereits viele fertige Importer für verschiedenste Daten wie zum Beispiel Arztbriefe, Laborwerte oder EKG-Ergebnisse.
Auch i2b2 bietet verschiedene einfachere Möglichkeiten zum Importieren von Daten:

- Für die i2b2 Workbench existiert das oben bereits vorgestellte *Import Data Plugin*. Hiermit können CSV-Dateien importiert werden, die allerdings schon über analoge Spalten zu der CRC-Tabellenstruktur verfügen müssen. Hierbei können dann lediglich die CSV-Spalten ihren entsprechenden Datenbankspalten zugewiesen werden. Außerdem lassen sich damit nur Daten für die Fakten- sowie für die Mapping-Tabellen importieren. Allerdings können die Fakten Patienten- und Fallkennungen aus anderen Systemen verwenden, die dann automatisch in entsprechende i2b2-Kennungen inklusive der jeweiligen Zuweisung in den Mapping-Tabellen umgewandelt werden.
- Die XML-Schnittstelle der CRC-Komponente unterstützt einen sogenannten „Publish Data Request“. Hierfür muss zunächst eine XML-Datei mit den zu importierenden Informationen im oben beschriebenen PDO-Format angelegt

werden. Auch hierbei können Patienten- und Fallkennungen aus anderen Systemen genutzt werden, die dann während des Importvorgangs umgewandelt werden. Die so erzeugte Datei wird entweder direkt im lokalen Dateisystem des i2b2-Servers oder alternativ mithilfe der FR-Zelle gespeichert. Der dabei verwendete Dateipfad muss dann als Teil der Anfrage angegeben werden. Mit dieser Methode können zusätzlich zu der Fakten- und den Mappingtabellen noch sämtliche Dimensionstabellen befüllt werden. Diese Methode ist allerdings gerade für größere Datenmengen relativ langsam.

– Als Teil des „Integrated Data Repository Toolkit“ gibt es ein *Import and Mapping Tool* (IMT) [42]. Dies ist eine Java-basierte Anwendung mit graphischer Benutzeroberfläche zum Import neuer Daten in i2b2. Die Oberfläche des Tools ist in Abbildung 10 zu sehen. Hierbei können die Daten in verschiedenen Formaten vorliegen. Unter anderem werden CSV-Dateien und ein spezielles XML-Format unterstützt. Auch der direkte Import aus einer anderen Datenbank ist möglich. Zusätzlich können Konzepte in verschiedenen Formaten separat oder in Verbindung mit Fakten importiert werden. Nach einem solchen Import ist es mit dem Tool dann außerdem möglich, die Konzepte beliebig anzupassen und die importierten Fakten entsprechend zu aktualisieren. Dabei können zum Beispiel mehrere Konzepte zusammengefasst oder auch die Konzepthierarchie verändert werden.

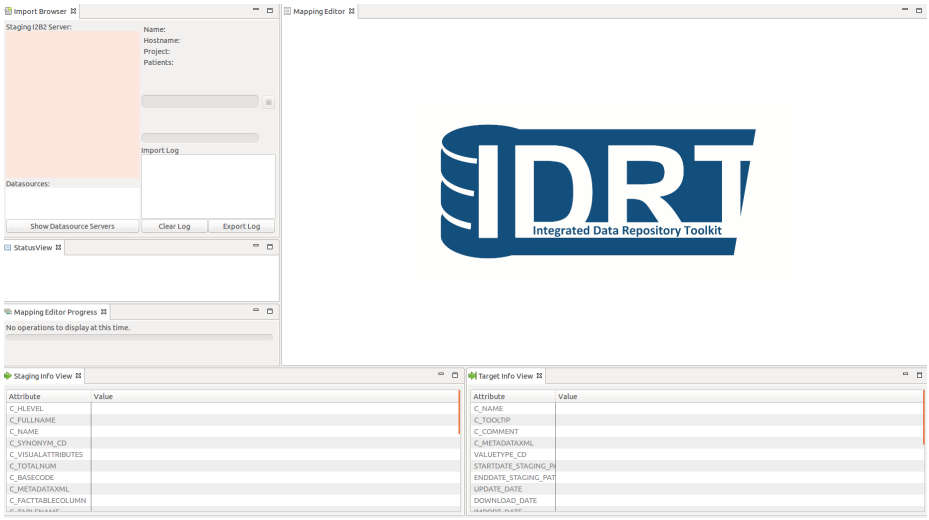

Abb. 10. Oberfläche des IDRT Import and Mapping Tools (entnommen von https://community.i2b2.org/wiki/display/IDRT/331.+General - aufgerufen am 23.03.2017, 16:00 Uhr)

4.3 Möglichkeiten der Anfragesprachen

Sowohl i2b2 als auch der PaDaWaN verwenden eine XML-basierte Anfragesprache. Während diese im PaDaWaN in Form von MXQL separat vom Rest des Systems auftritt, ist sie in i2b2 nur ein Teil der möglichen Anfragen an die CRC-Komponente. Ein grundlegender Unterschied dazwischen entsteht dadurch, dass von i2b2 Suchanfragen nicht direkt die gefundenen Werte sondern nur je nach Einstellung die Anzahl an Treffern sowie eine Verteilung der Treffer auf einzelne vordefinierte Attribute wie Alter und Geschlecht zurückgegeben wird. Um konkrete Werte abzufragen muss bei i2b2 zunächst eine Menge an passenden Patienten oder Fällen gesucht werden. Anschließend erhält man eine Kennung dieser Menge, mit der dann wiederum einzelne Daten zu Elementen in dieser Menge abgefragt werden können. Das, was im Folgenden zu den Möglichkeiten von i2b2-Anfragen geschrieben wird, bezieht sich auf die Verbindung dieser beiden Arten von Suchanfragen, da nur manche Möglichkeiten in beiden Arten vorhanden sind. Im PaDaWaN hingegen wird mit einer einzelnen Anfrage die Suche selbst und die gewünschte Rückgabe von Werten angegeben.
Viele der Anfragemöglichkeiten werden von beiden System unterstützt:

- Es können jeweils einzelne Attribute oder auch ganze Teile der Anfrage auf einen fixen Zeitraum oder auch nur auf Werte ab beziehungsweise bis zu einem bestimmten Zeitpunkt eingeschränkt werden.
- Es ist möglich zwischen beliebigen Teilen der Anfrage eine gewünschte zeitliche Relation festzulegen.
- Zeichenketten und numerische Werte können mit verschiedenen Operatoren wie zum Beispiel *enthält* (bei Texten) oder *kleiner gleich* (bei Zahlen) eingeschränkt werden.
- Die Suche kann darauf beschränkt werden, nur die Anzahl aller Treffern oder bei der Abfrage von Werten eine gewisse maximale Anzahl an Treffer zurückzugeben.
- Es kann festgelegt werden, dass Attribute pro Patient mindestens eine gewisse Anzahl mal vorkommen müssen, damit dieser Patient als Treffer zählt. Diese Funktion der Anfragesprache ist allerdings zur Zeit noch von keiner der PaDaWaN-Engines implementiert.
- Die Elemente einer Anfrage können mit den logischen Operatoren *AND*, *OR* und *NOT* verknüpft werden.
- Die Suche lässt sich auf einzelne fest vorgegebene Patienten- und Fallkennungen einschränken.
- Frühere Suchen können komplett als Teil einer neuen Anfrage genutzt werden.

Neben diesen in beiden Systemen sehr ähnlich vorkommenden Funktionen, gibt es auch einige, die unterschiedlich funktionieren:

- Im PaDaWaN kann genau festgelegt werden, ob für gefundene Elemente die Patienten-, Fall- oder Dokumentenkennung, der eigentliche Wert (oder optional nur ein „x“ für den Wert) und der Messzeitpunkt zurückgegeben

werden soll. Die einzelnen Rückgaben lassen sich außerdem noch über *distinct* und *multipleRows* genauer steuern. In i2b2 kann hier nur festgelegt werden, dass entweder nur die Spalten, die den Primary Key des Fakts bilden oder alle Spalten abgefragt werden sollen. Lediglich die Rückgabe des *OBSERVATION_BLOB* lässt sich separat steuern. In i2b2 kann zusätzlich festgelegt werden, dass auch die passenden Elemente einzelner Mapping- und Dimensionstabellen ausgegeben werden sollen.

- Während sich der *reductionOp* im PaDaWaN pro Attribut festlegen lässt, kann man in i2b2 nur eine einzelne Einschränkung der Rückgabewerte pro Anfrage verwenden.
- Im PaDaWaN steht mit der Verteilungssuche eine sehr flexible Möglichkeit zur Verfügung, um sich die gefundene Anzahl von Patienten aufgeteilt auf einzelne Jahre oder auch auf die direkten Nachfolger eines beliebigen Eintrags in der Kataloghierarchie ausgeben zu lassen. i2b2 bietet hier lediglich die vier fixen Aufteilungen auf Altersgruppen, den Wert von *VITAL_STATUS_CD*, das Geschlecht und die Hautfarbe.
- In i2b2 kann angegeben werden, dass gewisse Teile der Anfrage im selben Fall oder mit der selben *INSTANCE_NUM* vorkommen sollen. Die Ausgabe ist aber trotzdem immer die Anzahl an Patienten. Im PaDaWaN kann mit diesen Einstellungen die komplette Gruppierung der Rückgabe auf Fall- beziehungsweise Dokumenten-Ebene abgeändert werden.
- Nur mit MXQL lassen sich relative Suchen über gefundene Werte definieren.

Durch die zusätzlich bei i2b2-Fakten gespeicherten Informationen, gibt es dort auch spezielle Einschränkungen für Modifier und für die *VALUEFLAG_CD*. Wie im vorherigen Abschnitt schon erläutert, lassen sich diese Funktionen beide anders im PaDaWaN modellieren (Modifier als „normale“ Katalogeinträge und die *VALUEFLAG_CD* lässt sich anhand der oberen und unteren Grenze für numerische Werte ermitteln) und sind deswegen dort nicht erforderlich. Zeitlich relative Suchen in i2b2 können neben dem *START_DATE* auch über dem *END_DATE* ausgeführt werden. Da es ein solches bei den PaDaWaN-Informationen nicht gibt, fehlt dort diese Möglichkeit.
Während somit nahezu alle i2b2-Suchfunktionen vom PaDaWaN unterstützt werden, gibt es einiges, das nur von der PaDaWaN-Anfragesprache MXQL unterstützt wird. So ist die Funktion *searchSubEntries* in i2b2 immer aktiviert, während sie sich im PaDaWaN pro Attribut einstellen lässt. Eine Umkreissuche in Texten ist mit i2b2 genauso wenig möglich, wie eine relative Suche über einzelne Werte. Auch der *SortOperator* und die Parameter *maxCount*, *countType* und *valueInFile* (die allerdings teilweise noch von keiner PaDaWaN-Engine implementiert wurden) werden nicht unterstützt. Außerdem fehlt in i2b2 die Möglichkeit zum Hervorheben von gesuchten Wörtern innerhalb eines gefundenen Textes sowie die spezielle Suche nach im positiven oder negierten Kontext vorkommenden Begriffen.

4.4 Möglichkeiten der Oberflächen

Für beide Systeme steht sowohl eine Web- als auch eine native Anwendung mit graphischer Benutzeroberfläche zum Zugriff auf sowie zum Durchsuchen der Daten zur Verfügung. Der grundlegende Aufbau ist dabei immer ähnlich.
Standardmäßig gibt es jeweils im linken Bereich eine Möglichkeit zum Betrachten sowie zum Durchsuchen der Kataloghierarchie. Aus dieser heraus lassen sich dann die einzelnen Konzepte in neue Anfragen übernehmen.
Die Anfrage selbst wird jeweils im oberen Bereich der Oberfläche zusammengebaut. Dabei lassen sich sich ein Großteil der Parameter der Anfragesprachen direkt einstellen. In beiden i2b2-Oberflächen passiert dies hauptsächlich durch ein Sortieren der Elemente in einzelne Panels sowie über Kontextmenüs der Konzepte. In der RCP-Oberfläche des PaDaWaN hingegen werden die Parameter in Tabellenform mit verschiedenen Checkboxen, Drop-Down-Menüs und Textfeldern festgelegt. Die PaDaWaN-Weboberfläche verwendet eine eigene Anfragesprache zum definieren der Suchen für jedes einzelne Element.
Im unteren Teil der Oberflächen wird dann jeweils das Ergebnis der Anfrage angezeigt. Dies unterscheidet sich im Fall des PaDaWaN je nachdem, ob eine Patienten- oder eine Verteilungssuche ausgeführt wurde. Auch bei i2b2 wird die Anzeige je nach den gewählten Rückgaben (also zum Beispiel *PATIENT_COUNT_XML* oder *PATIENT_ENCOUNTER_SET*) angepasst.
In allen Oberflächen gibt es weiterhin die Möglichkeit, bestehende Anfragen zu speichern und frühere Anfragen zu verwalten sowie diese zu laden und als Teil einer neuen Anfrage zu benutzen. Die Anfragen werden von i2b2 automatisch gespeichert, was im PaDaWaN je nach Bedarf manuell gemacht werden. Da somit nicht alle letzten Anfragen in den PaDaWaN-Oberflächen zur Verfügung stehen, bieten diese jeweils eine Historie der letzten innerhalb einer Anmeldungssitzung eines Nutzers ausgeführten Suchen.
Eine weitere Gemeinsamkeit aller Oberflächen ist die Möglichkeit, Ergebnisse einer Anfrage zu exportieren.
Neben diesen Gemeinsamkeiten gibt es aber auch einige Unterschiede zwischen den Oberflächen:

- Während bei den PaDaWaN-Oberflächen direkt die gefunden Werte dargestellt werden können, ist dies in der i2b2-Weboberfläche nur mittels des „ExportXLS Analysis Tool“ und in der i2b2-Workbench standardmäßig gar nicht sondern nur über ein separat zu installierendes Plugin möglich.
- Die Darstellung der gefundenen Ergebnisse mittels einer Timeline ist nur in den i2b2-Oberflächen möglich. Allerdings wird eine ähnliche Möglichkeit für den PaDaWaN zur Zeit entwickelt.
- Verschiedene i2b2-spezifische Funktionen, wie die der IM- und WORK-Komponente sind dementsprechend nur in den i2b2-Oberflächen zu finden.
- Die verschiedenen Möglichkeiten der Verteilungssuchen im PaDaWaN fehlen hingegen bis auf die im Abschnitt zu der CRC-Komponente beschriebenen fixen Optionen zur Rückgabe von Verteilungen in i2b2.
- I2B2 bietet die Unterstützung für mehrere Analysis Tools in der Weboberfläche und Plugins für die Workbench. Neben bereits bestehenden können

hier auch noch eigene ergänzt werden. Diese Option gibt es in keiner der PaDaWaN-Oberflächen.
- Nur in den i2b2-Oberflächen können relative temporale Suchen definiert werden.
- Ausschließlich in der i2b2-Workbench besteht die Möglichkeit Katalogeinträge direkt zu bearbeiten.
- Die RCP-Oberfläche des PaDaWaN bietet als einzige die Möglichkeit zur Einschränkung der Suche auf fixe Kennungen für Patienten und Fälle. In i2b2 ist hier lediglich die Einschränkung auf von früheren Suchen gefundene Mengen von Patienten beziehungsweise Fällen möglich.
- Während die Anzahl passender Patienten für die Konzepte in allen Oberflächen dargestellt werden können, ist dies für die Anzahl passender Fälle sowie für die sämtlicher passender Informationen nur in den PaDaWaN-Oberflächen möglich, da diese Werte in der i2b2-Datenstruktur gar nicht vorkommen.
- Nur über die i2b2-Oberflächen kann der angemeldete Benutzer sein Passwort selber ändern.

Ein großer Unterschied zwischen den Oberflächen besteht auch noch darin, wie auf die Systeme im Hintergrund zugegriffen wird.
Nur in der RCP Oberfläche des PaDaWaN werden dazu in einer Konfigurationsdatei sämtliche Zugänge für die Datenbank, eventuell verwendete Authentifizierungs- und Autorisierungsserver sowie die Zugänge für die genutzten Engines gespeichert. Das erlaubt ein relativ einfaches Anpassen dieser Konfiguration und ermöglicht der Oberfläche den direkten Zugriff auf die Datenquellen.
In i2b2 hingegen und auch in der Weboberfläche des PaDaWaN wird in der Konfigurationsdatei nur die Adresse der i2b2-PM-Komponente beziehungsweise der Webschnittstelle des PaDaWaN festgelegt. An diese wird dann die Anmeldung des Nutzers geschickt, welche anschließend selbst eventuell genutzte Authentifizierungs- und Autorisierungsserver abfragt sowie die Adressen der für das gewünschte Projekt zu verwendenden anderen Komponenten zurückgibt. Dies hat den Vorteil, dass die Oberflächen selbst nie direkt auf die Datenquellen zugreifen müssen (und dies Aufgrund der ihnen gar nicht bekannten Zugängen (zum Beispiel Datenbankzugängen) auch gar nicht können), sondern stattdessen alle in der Oberfläche dargestellten Daten mithilfe der XML-Schnittstellen abfragen. Im Falle von i2b2 fragen auch die Plugins wie zum Beispiel das *ExportXLS Analysis Tool* die Daten immer zunächst mithilfe dieser Schnittstellen ab und machen ihre Auswertungen dann erst über den so erhaltenen Informationen.

4.5 Interne Anfragenverarbeitung

Sobald eine Suchanfrage an eines der Systeme gestellt wird, wird sie von diesem in mehreren Schritten verarbeitet.
Im PaDaWaN sind das die folgenden:

- Zunächst muss eine Instanz eines sogenannten *GuiQueryClient* erzeugt beziehungsweise eine vorhandene verwendet werden. Diese legt fest, auf welcher Engine die Anfrage ausgeführt wird.

- Optional kann dann zunächst geprüft werden, ob die gewünschte Suche überhaupt mit der aktuell verwendeten Engine möglich ist.
- Die eingelesene MXQL-Anfrage wird dann dem *QueryRunner* des *GuiQueryClient* übergeben, der zunächst für den Nutzer, der die Anfrage stellt die Parameter abruft, die festlegen, ob dieser Nutzer die Anfrage überhaupt ausführen und die Ergebnisse angezeigt bekommen darf.
- Der *QueryRunner* erzeugt wiederum ein *QueryRunnable* und gibt dessen zugehörige Kennung zurück.
- Mit dieser Kennung kann der *QueryRunnable* dann abgerufen und mit ihm wiederum die eigentliche Ausführung der Anfrage gestartet werden.
- Nach dem Starten der Ausführung wird zunächst in einem Zwischenspeicher überprüft, ob die Anfrage dort bereits vorhanden ist. In diesem Fall wird ein gespeichertes Ergebnis zurückgegeben, ohne die Suchengine selbst überhaupt zu verwenden.
- Falls es eine neue Anfrage ist, so wird diese dann an die eigentliche Engine übergeben. Dies wandelt dann das MXQL der Suche in eine zur Engine passende Anfrage um und führt diese aus.
- Das Ergebnis der Engine muss dann von dieser in pro gefundenem Patient, Fall oder Dokument jeweils eine Liste mit sogenannten *ResultCellData*-Objekten umgewandelt werden. Ein solches Objekt enthält jeweils für ein Attribut der Anfrage alle zum aktuellen Treffer passenden Informationen.
- Für jeden Treffer findet dann je nach Engine noch eine Nachbearbeitung der Ergebnisse statt, falls die verwendete Engine in der Anfrage verwendete Elemente nicht direkt sondern nur mithilfe dieser Nachbearbeitung unterstützt. Dies kann zum Beispiel für die relativen Suchen gemacht werden und löscht dann dazu nicht passende Treffer aus der Ergebnismenge.
- Abschließend werden die übergebenen Treffer in Abhängigkeit von den Parametern der Anfrage in eine Tabellenstruktur einsortiert, welche dann als Ergebnis dieser Anfrage zurückgegeben wird.

i2b2 verarbeitet in seiner Standardimplementierung die erhaltenen Suchanfragen auf die im Folgenden anhand von [43] beschriebene Art und Weise:

- Zunächst wird die Anmeldung des aktuellen Benutzers und dessen Berechtigung zur Ausführung der gestellten Anfrage mithilfe der PM-Zelle überprüft.
- Anschließend wird die zu verwendende Datenquelle mithilfe der übergebenen Domain-, Projekt- und Benutzerkennung ermittelt.
- Für alle Attribute der Anfrage wird dann die ONT-Komponente mit dem Pfad des jeweiligen Attributs abgefragt, um die für das zugehörige Konzept zu verwendende und mit der Faktentabelle zu verknüpfende Dimensionstabelle abzufragen.
- Mit dieser Information wird dann eine SQL-Anfrage generiert, die zusammen mit der XML-Anfrage als sogenannter *QueryMaster* gespeichert wird.
- Zur Ausführung der Anfrage wird dann eine *QueryInstance* erzeugt, in welcher der aktuelle Status der Anfrage gespeichert ist.
- Um auch Anfragen zu unterstützen, die länger laufen und dabei trotzdem die Stabilität des Systems zu gewährleisten, gibt es drei sogenannte *Queues*,

mit denen die Ausführung der Anfragen verwaltet wird. Zunächst ist eine Anfrage dabei immer in der *Small Job Queue*. Sobald sie länger als ein bestimmter Grenzwert läuft, wird sie in die *Mid Job Queue* und wenn sie auch dort noch länger braucht in die *Large Job Queue* verschoben.
- Als Teil der Anfrage wird immer mit angegeben, wie lange auf eine Rückmeldung gewartet werden soll. Wird dieser Zeitraum überschritten, so wird anstelle der Ergebnisse der Anfrage deren aktueller Status zusammen mit einem Zeitfenster, nachdem dieser Status wieder abgefragt werden soll, zurückgegeben.
- Sobald die Anfrage abgeschlossen ist, wird für jede gewünschte Rückgabe eine *QueryResultInstance* erzeugt. Diese verweist dann wiederum auf die eigentlichen Ergebnisse, die jeweils nochmal separat in eigenen Tabellen gespeichert werden. Anhand der Kennung der einzelnen *QueryResultInstances* lassen sich diese Ergebnisse dann auch im Nachhinein noch abfragen und anzeigen.
- Als letztes werden dann alle Ergebnisse zurückgegeben.
- Wenn dann noch konkrete Werte zu den gefundenen Patienten oder Fällen mittels einer PDO-Anfrage abgefragt werden, werden dabei zunächst die ersten drei der obigen Schritte wiederholt. Für diese Anfragen wird nur das zugehörige XML-Dokument gespeichert und anschließend direkt die Werte aus der Datenbank abgefragt und in PDO-XML-Form zurückgegeben.

Wenn beide Systeme miteinander verbunden werden sollen, so muss dazu somit immer zunächst die Anfrage in das Format des jeweils anderen Systems umgewandelt und an dieses gestellt werden. Nach der Ausführung der Anfrage findet dann wieder eine Umwandlung in die andere Richtung mit den zurückgegeben Ergebnissen statt. Wie genau diese Ergebnisse in beiden Systemen jeweils aussehen, wird im folgenden Abschnitt genauer beschrieben. Zu beachten ist bei der Verbindung der Systeme außerdem, dass für eine Rückgabe von Werten zwei separate i2b2-Anfragen (Patient-Set-Anfrage und PDO-Anfrage) nötig sind, während dies mit nur einer PaDaWaN-Anfrage erledigt werden kann.

4.6 Rückgaben

Hinsichtlich der Rückgabe von Anfrageergebnissen gibt es ebenfalls einige Unterschiede zwischen i2b2 und dem PaDaWaN.
Im PaDaWaN gibt es zwei verschiedene Arten von Rückgaben. Das ist einerseits die der Verteilungs- und andererseits die der „normalen“ Suche. Beide geben ihre Werte jedoch in einer Tabellenstruktur zurück:

- Im Rahmen der Verteilungssuche können mehrere Attribute festgelegt werden, für deren Treffer dann zum Beispiel eine Verteilung über mehrere Jahre ausgegeben wird. Jedes dieser Attribute erzeugt eine eigene Zeile in der Rückgabe. In den Spalten steht dann jeweils die Anzahl an Suchtreffern pro zum Beispiel Jahr, sowie die Summe aller Treffer.

– Bei der Suche nach konkreten Werten wird in der Regel pro gefundenem Patient, Fall oder Dokument eine Zeile im Ergebnis erzeugt. Die Spalten entstehen dann je nachdem welche Paramater in der Anfrage benutzt wurden. So kann es dann eine Spalte mit der Patientenkennung, sowie pro Attribut jeweils eine Spalte mit dessen Wert, Fall- und Dokumentenkennung sowie Messzeitpunkt geben. Werden für eine Zeile dabei mehrere Werte gefunden so werden diese, falls kein *reductionOp* verwendet wurde, standardmäßig mit Kommata getrennt. Alternativ kann auch der Parameter *multipleRows* genutzt werden, was dann bewirkt, dass bei mehreren Werten pro Treffer für jeden dieser Werte eine separate Zeile in der Ergebnisstabelle angelegt wird. Auch bei einer reinen *onlyCount*-Anfrage wird eine Tabelle, dann allerdings nur mit einer Zeile und Spalte, erzeugt. In diesem einen Feld steht dann nur die gefundene Anzahl an Treffern.

Während bei i2b2 zwar einige der Plugins wie zum Beispiel das *ExportXLS Analysis Tool* die Ergebnisse auch in tabellarischer Form aufbereitet anzeigen, so ist die eigentliche Rückgabe von i2b2 immer in XML-Form. Für die sogenannte *Patient Set*-Anfrage gibt es die folgenden Rückgabemöglichkeiten:

– Es kann sowohl die Menge an passenden Patienten als auch die an passenden Fällen abgefragt werden. Die Rückgabe hiervon umfasst jedoch nur eine Kennung der zugehörigen *QueryResultInstance* sowie die Anzahl an Elementen in dieser Menge. Um sich dann die zugehörigen Fakten oder auch die Werte der Patienten- beziehungsweise Fall-Dimensionstabellen ausgeben zu lassen, muss mit dieser Kennung eine separate *Patient Data Object*-Anfrage gestellt werden, deren Rückgabe weiter unten noch beschrieben wird.
– Zusätzlich gibt es verschiedene spezielle XML-Rückgaben. Auch hierbei erhält man zunächst nur die Kennung der *QueryResultInstance* und fragt dann separat das eigentliche XML-Ergebnis mit einer entsprechenden CRC-Anfrage ab. Als mögliche Rückgaben gibt es hier neben der reinen Anzahl an zur Anfrage passenden Patienten noch eine Aufteilung dieser Anzahl auf entweder das Geschlecht, die Altersgruppe, die Hautfarbe oder den Wert von *VITALSTATUS_CD* der gefundenen Patienten.

Wie bereits erwähnt, werden konkrete Werte in i2b2 nur von sogenannten *Patient Data Object*-Anfragen zurückgegeben. Hierbei werden je nach Parametern der Anfrage die Werte zu einer festgelegten Menge von Patienten oder Fällen sowie zu einzelnen Attributen abgefragt. Außerdem legt die Anfrage die Tabellen der CRC-Zelle fest, deren Werte abgefragt werden sollen. Dabei ist außerdem immer noch die Einschränkung auf eine ausschließliche Rückgabe der Primary Keys der Tabellen oder aller darin gespeicherter und zur Anfrage passender Werte möglich. Die Rückgabe ist dann ein PDO-XML-Dokument, dessen Struktur genau dem weiter oben beschriebenen Standardaufbau der CRC-Tabellen entspricht.
Wie im letzten Abschnitt bereits erwähnt ist bei einer Verbindung der beiden Systeme folglich jeweils eine Umwandlung der Rückgaben in das Format des jeweils anderen Systems erforderlich. Dabei müssen dann entsprechend auch die zuvor bereits erläuterten Unterschiede in der Datenstruktur berücksichtigt werden.

4.7 Nutzer- und Projektmanagement

Um sowohl i2b2 als auch den PaDaWaN nutzen zu können, benötigt man einen Login aus Benutzernamen und Passwort. Diese Nutzer wiederum haben dann gewisse Rollen und damit verbundene Rechte, die festlegen, welche Funktionen sie benutzen und damit verbunden auch, welche Formen von Anfragen sie stellen dürfen.
Hierbei gibt es zwischen den Systemen einige Unterschiede und Gemeinsamkeiten:

- In beiden Systemen gibt es Administratoren, die die restlichen Nutzer und deren Rechte verwalten können, sowie „normale“ Benutzer des Systems mit wiederum unterschiedlichen Rechten.
- Beide Systeme bieten die Möglichkeit, dass die Nutzer und deren Rechte direkt in den jeweiligen Datenbanken oder alternativ auch auf externen Authentifizierungs- und Autorisierungsserver liegen können. Während das im PaDaWaN alles einzeln in der Konfigurationsdatei hinterlegt werden muss, wird es in i2b2 zentral mithilfe der PM-Komponente verwaltet.
- In i2b2 können Nutzer im Gegensatz zum PaDaWaN mehreren Projekten zugeordnet sein und in diesen jeweils auch unterschiedliche Rechte haben. Für jedes Projekt können dann unterschiedliche Adressen für die anderen i2b2-Komponenten hinterlegt werden. In diesem Zusammenhang gibt es in i2b2 dann noch die Rolle des Managers, welcher ebenfalls andere Nutzer verwalten kann, was im Gegensatz zum Administrator allerdings auf einzelne Projekte beschränkt ist. Die Konfiguration der Projekte selbst kann nur von Administratoren vorgenommen werden. Projekte oder etwas ähnliches gibt es im PaDaWaN nicht. Hier könnten lediglich mehrere komplette und separate Instanzen aufgesetzt werden.
- Sämtliche Nutzerrechte werden im PaDaWaN mithilfe von Gruppen konfiguriert. Jeder Nutzer sollte dabei mindestens einer Gruppe zugeordnet sein, da er andernfalls keine Katalogeinträge in den Oberflächen angezeigt bekommt und somit auch keine Suchen nach Fakten erstellen kann. Er kann aber auch mehreren Gruppen zugewiesen werden. Während diese Gruppenzugehörigkeit im PaDaWaN sowohl die eventuellen Administratorrechte als auch die Rechte zum Zugriff auf die vorhandenen medizinischen Daten festlegt, gibt es in i2b2 speziell für den Datenzugriff neben den bereits beschriebenen Rollen noch sogenannte *Data Protection Roles*, die im Abschnitt zur PM-Komponente genauer erläutert wurden. Während diese allerdings fix sind, können im PaDaWaN beliebige Gruppen mit beliebigen Rechten angelegt werden.
- In beiden Systemen ist es mit obigen Zuordnungen möglich den Zugriff auf ausschließlich aggregierte Daten (im PaDaWaN also auf die Verteilungssuche) einzuschränken. Nur in i2b2 kann dies inklusive einer optionalen Beschränkung der maximalen Anzahl, die dieselbe Anfrage in einem bestimmten Zeitraum ausgeführt werden darf, gemacht werden. Nur im PaDaWaN hingegen lässt sich ein Grenzwert einstellen, sodass die Ergebnisse einzel-

ner Teilanfragen einer Verteilungssuche zensiert werden können, wenn sie unterhalb dieses Grenzwertes liegen.

- Wird die obige Option nicht genutzt, so können die Nutzer in beiden Systemen auch konkrete Werte abfragen. Nur in i2b2 gibt es hier noch zusätzlich die Möglichkeit dabei verschlüsselte Daten auszuschließen, welche es im PaDaWaN allerdings gar nicht gibt, da in diesen zur Zeit nie vertrauliche und somit zu verschlüsselnde Daten importiert werden.
- i2b2 verfügt außerdem noch über eine *Data Protection Role*, mit der auch auf die Inhalte der IM-Komponente zugegriffen werden kann. Dies ist die Rolle mit den meisten Rechten und umfasst somit auch vollen Zugriff auf alle anderen Daten.
- Der PaDaWaN erlaubt durch Black- und Whitelisting (was bei der Beschreibung der Datenbankstruktur des PaDaWaN bereits erläutert wurde) eine sehr flexible Konfiguration der Fälle sowie der Teile der Kataloghierarchie, die für die Nutzer der entsprechenden Gruppe in der Oberfläche sichtbar sind. Die Möglichkeit zur Einschränkung der sichtbaren Fälle gibt es in i2b2 nicht. Der Katalog kann zwar eingeschränkt werden (vergleiche [44]), allerdings nur in so fern, dass einzelne Elemente auf der höchsten Hierarchieebene (inklusive aller untergeordneten Elemente) nur den Nutzern mit der *Data Protection Role* mit den meisten Rechten angezeigt werden.
- Für Administratoren (und in i2b2 auch für Manager) gibt es in beiden Systemen eine Weboberfläche, über die sich ein Großteil der in den vorherigen Punkten beschriebenen Funktion konfigurieren lassen. Die Administrationsoberfläche von i2b2 ist in Abbildung 11 und die des PaDaWaN in Abbildung 12 zu sehen.

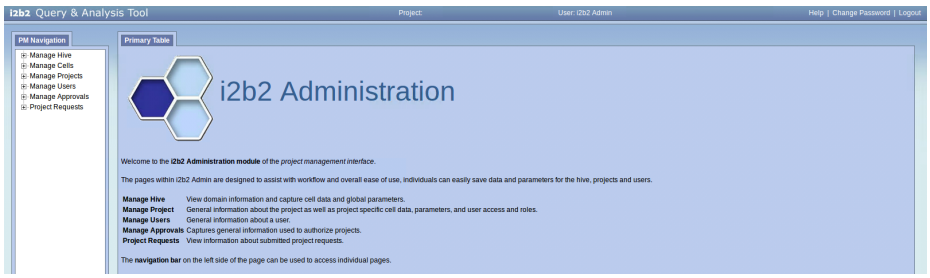

Abb. 11. Administrationsoberfläche von i2b2 (eigener Screenshot)

Bei einer Verbindung beider Systeme gibt es in diesem Punkt folglich einige Schwierigkeiten. Vor allem die fehlende Möglichkeit für separate Projekte im PaDaWaN sowie die in i2b2 nicht beziehungsweise nur sehr eingeschränkt vorhandene Option zur Beschränkung der verwendbaren Fälle und Katalogeinträge stellen dabei Probleme dar.

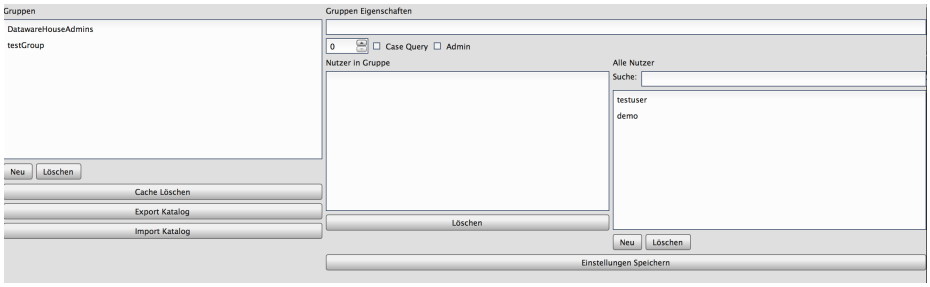

Abb. 12. Administrationsoberfläche des PaDaWaN (eigener Screenshot)

5 Kopplung von i2b2 und PaDaWaN

In diesem Kapitel wird der im Rahmen dieser Arbeit entstandene Code[6] genauer beschrieben.

5.1 Datentransfer zwischen i2b2 und PaDaWaN

Zunächst wurde eine Möglichkeit entwickelt, mit der Daten zwischen den beiden Systemen übertragen werden können.
Zur Verwaltung sämtlicher Daten des PaDaWaN existieren die beiden Klassen *CatalogManager* und *InfoManager*. Mit diesen ist ein Abfragen und Exportieren sämtlicher Daten, sowie das Löschen bestehender und importieren neuer Daten möglich. Um all dies nun auch für die Daten innerhalb von i2b2 zu ermöglichen, wurde ein *I2B2CatalogManager* sowie ein *I2B2InfoManager* entwickelt, der jeweils von den entsprechenden PaDaWaN-Klassen erbt und somit sämtliche Funktionen dieser Klassen auch für i2b2 unterstützt. Dadurch können für einen Datentransfer dann die Daten mithilfe eines Managers ausgelesen beziehungsweise exportiert und mit dem entsprechenden des anderen Systems wieder importiert werden.
Die Manager selber verwenden zum Zugriff auf die eigentlichen Daten sogenannte *Adapter*. Diese bestehen für die PaDaWaN-Manager bereits für die davon unterstützten Datenbanktypen (MSSQL und MySQL) und mussten für die i2b2-Manager nun ebenfalls gebaut werden.

5.1.1 I2B2CatalogManager

Der CatalogManager basiert auf insgesamt 4 Adaptern. Der wichtigste davon ist der *CatalogAdapter*. Dessen Funktionen wurden für i2b2 wie folgt implementiert:

[6] Komplett einsehbar nach entsprechender Freigabe unter https://gitlab2.informatik.uni-wuerzburg.de/misbased/misbased-sandbox/tree/develop/misbased-dw-query-i2b2-new und unter https://gitlab2.informatik.uni-wuerzburg.de/misbased/misbased-sandbox/tree/develop/I2B2Rest (die Pfade sind die, die zur Zeit des Verfassens dieser Arbeit verwendet wurden und können sich eventuell später ändern)

– Bei seiner Initialisierung legt der *I2B2CatalogAdapter* zunächst eine Verbindung zur i2b2-Datenbank an. Diese lässt sich über die folgenden Parameter in der Konfigurationsdatei konfigurieren:
  - *dw.index.i2b2.db.type* legt den Typ der Datenbank fest. Unterstützt werden dabei im Moment eine MSSQL- sowie eine Oracle-Datenbank. Dieser Parameter muss dann entweder den Wert „MSSQL“ oder „ORACLE“ haben.
  - Mit *dw.index.i2b2.db.address* und *dw.index.i2b2.db.port* werden Adresse und Port der Datenbank festgelegt.
  - *dw.index.i2b2.db.ont.database* gibt den Namen der Datenbank an.
  - *dw.index.i2b2.db.ont.user* und *dw.index.i2b2.db.ont.password* legen Benutzernamen und Passwort zum Zugriff auf die Datenbank fest.
– Um mithilfe des CatalogManager einen schnellen Zugriff auf den Katalog zu ermöglichen, werden beim Erstellen einer neuen Instanz immer zunächst sämtliche aktuell vorhandene Einträge in den Arbeitsspeicher geladen. Zu diesem Zweck wird als erstes die *readTables*-Methode des *CatalogAdapter* aufgerufen. In dieser Methode werden die folgenden Schritte durchgeführt:
  - Als erstes wird die *CUSTOM_META*-Tabelle abgefragt. Dies ist in der Standardimplementierung von i2b2 die Katalogtabelle, unter der benutzerdefinierte Katalogeinträge hinzugefügt werden können. Dort werden bei einem Import eines PaDaWaN-Katalogs in i2b2 die übernommen Einträge gespeichert, weswegen dieser Schritt dazu dient, den *I2B2CatalogManager* auch für in i2b2 übertragene PaDaWaN-Daten verwenden zu können. Falls diese Tabelle einen anderen Namen hat, so lässt sich dieser Name mit dem Konfigurationsparameter *dw.index.i2b2.db.ont.custTableName* anpassen. Bei der Abfrage werden sämtliche Zeilen der Tabelle in der Reihenfolge ihrer Hierarchie abgerufen (also aufsteigend sortiert nach dem Wert der *C_HLEVEL*-Spalte).
  - Mit jeder so gefundenen Tabellenzeile wird dann eine Instanz der *I2B2OntologyEntry*-Klasse erstellt. Diese dient unter anderem dazu, eine solche Zeile in eine Instanz eines PaDaWaN-*CatalogEntry* umzuwandeln. Wie das genau gemacht wird, wird weiter unten noch beschrieben.
  - Der so generierte *CatalogEntry* wird noch dem in der Kataloghierarchie direkt über ihm stehenden Eintrag zugeordnet und anschließend im CatalogManager gespeichert.
  - Während dem Laden dieser aus PaDaWaN übernommenen Eintrag wird außerdem noch die insgesamt höchste *AttrID* gespeichert, damit für im folgenden geladene i2b2-Einträge keine bereits verwendete *AttrID* erneut benutzt wird.
  - Ist nun der Konfigurationsparameter *dw.index.i2b2.limitToPadawan* auf „true“ gesetzt, so ist das Laden des CatalogManager abgeschlossen. Dieser Parameter legt nämlich fest, dass sich die i2b2-Klassen auf aus dem PaDaWaN übernommene Daten beschränken sollen. Steht er hingegen auf „false“, so werden auch noch i2b2-eigene Daten geladen.
  - Dafür werden dann sämtliche mit dem Parameter *dw.index.i2b2.db.ont.otherTableNames* festgelegte Tabellen mit der selben Sortierung der Ergebnisse wie oben abgefragt. Werden hier mehrere Tabellen angegeben,

so müssen diese in der Konfigurationsdatei jeweils durch ein „#“ getrennt werden.
    - Auch hier ist wieder die *I2B2OntologyEntry*-Klasse für das Umwandeln der Tabellenzeilen in *CatalogEntry*-Elemente zuständig.
    - Da in i2b2 manche Elemente der PaDaWaN-Katalogeinträge nicht vorhanden sind, werden diese anschließend noch generiert. Als *AttrID* erhält ein Eintrag den Wert der eins über der aktuell höchsten *AttrID* liegt. Als Elternelement wird dann in Abhängigkeit des Wertes von *C_HLEVEL* entweder der *Root-CatalogEntry* oder der dann bereits eingelesene Elterneintrag gesetzt. Außerdem wird noch der *UniqueName* mithilfe der entsprechenden CatalogManager-Methode generiert. Der so erzeugte *CatalogEntry* wird dann im CatalogManager selbst (im Arbeitsspeicher) gespeichert.
- Beim Importieren von PaDaWaN-Einträgen in i2b2 wird zunächst der Elterneintrag abgefragt und dem neuen Eintrag zugewiesen. Der Elterneintrag wird außerdem benötigt, da in i2b2 für jeden Eintrag gespeichert wird, ob dieser noch untergeordnete Elemente hat, oder nicht, da dies dann in den Benutzeroberflächen anders dargestellt wird. Die die Einträge einzeln importiert werden, wird jeder neue Eintrag erstmal so importiert, als hätte er keine Kinderelemente. Wenn nun der abgerufene Elterneintrag noch nicht für untergeordnete Elemente angepasst wurde, so wird dies vor dem Import des neuen Eintrags als erstes gemacht. Anschließend wird wieder mithilfe der *I2B2OntologyEntry*-Klasse der neue *CatalogEntry* in das I2B2-Format umgewandelt und in der Datenbank gespeichert.
- Wenn ein Eintrag aktualisiert werden soll, so wird, um nicht jedes Feld einzeln auf die durchgeführte Veränderung überprüfen zu müssen, zunächst der bisherige Eintrag gelöscht und anschließend der neue Eintrag wieder gespeichert. Hierbei wird aktuell noch kein Verschieben von Einträgen innerhalb der Hierarchie unterstützt. Dies ist jedoch auch über die XML-Schnittstelle von i2b2 nicht möglich.
- Zum Abrufen eines einzelnen Eintrags verwendet der *CatalogAdapter* eine Kombination aus *ExtID* und *Project*. Diese werden bei aus dem PaDaWaN übernommenen Einträgen in den *C_BASECODE* kodiert, weswegen der Eintrag zunächst anhand dieser Spalte gesucht wird. Bei i2b2-eigenen Einträgen werden *ExtID* und *Project* mithilfe des Pfades eines Eintrags generiert (mehr dazu weiter unten). Deswegen wird, falls die Suche mithilfe des *C_BASECODE* erfolglos war, anschließend noch die *C_FULLNAME*-Spalte durchsucht.
- Soll ein einzelner Eintrag gelöscht werden, so wird dieser zunächst wieder mithilfe der *I2B2OntologyEntry*-Klasse in das i2b2-Format umgewandelt und anhand der so erhaltenen Informationen aus der Datenbank gelöscht.
- Beim Löschen aller Einträge werden dann in Abhängigkeit des Wertes von *dw.index.i2b2.limitToPadawan* entweder alle oder nur aus PaDaWaN übernommene Werte aus der Datenbank gelöscht.

Die bereits erwähnte *I2B2OntologyEntry*-Klasse wird zum Umwandeln der Einträge zwischen dem PaDaWaN- und dem i2b2-Format genutzt. Dazu kann sie

entweder mit einem PaDaWaN-*CatalogEntry* oder mit einem i2b2-Eintrag entweder in Form eines *ResultSet* von einer Datenbankabfrage, oder in Form eines XML-Dokumentes, welches von der XML-Schnittstelle der i2b2-ONT-Komponente verwendet wird, initialisiert werden. Damit diese Umwandlung funktioniert werden immer zunächst mithilfe des übergebenen Eintrags alle einzelnen Felder eines i2b2-Eintrags extrahiert beziehungsweise generiert und in der Instanz der Klasse gespeichert.
Eine Umwandlung eines PaDaWaN-Eintrages in einen i2b2-Eintrag, zum Beispiel um den PaDaWaN-Eintrag in einer i2b2-Datenbank zu speichern oder auch über die i2b2-Oberflächen auszugeben, funktioniert dabei mithilfe der folgenden Schritte:

- Anhand des Namens des aktuellen Eintrags und des der Elterneinträge wird der Pfad zusammengebaut. Dieser beginnt immer mit der Kennung für die *CUSTOM_META*-Tabelle und enthält dann jeweils durch Backslash-Zeichen getrennt die Namen aller Einträge auf dem Pfad bis hin zum aktuell betrachteten Eintrag, dessen Name den Pfad abschließt. Da der Pfad nur eine gewisse Maximallänge haben darf, werden Namen mit mehr als 20 Zeichen auf 20 gekürzt. Die letzten der dann verbliebenen 20 Zeichen werden durch „...“ ersetzt, um die Kürzung des Namens auch im Nachhinein erkennen zu können. Bei Informationen im i2b2-Format kann anstelle der anfänglichen Kennung für die *CUSTOM_META*-Tabelle auch eine andere verwendet werden, da diese Einträge in der Regel nicht in dieser Tabelle stehen. Dafür kann mithilfe des Parameters *dw.index.i2b2.ont.fullnameToTablecdMapFile* der Pfad zu einer Datei angegeben werden, die den Anfang des *C_FULLNAME* einer Tabellenkennung zuordnet. In dieser Datei muss dann pro Zeile eine Zuordnung in der Form „<*C_FULLNAME-Anfang*>=<*Tabellenkennung*>“ stehen. Eine zur Standardimplementierung passende Datei wird zusammen mit dem Code dieser Arbeit bereitgestellt.
- Beim Durchgehen der Hierarchie für den aktuellen Eintrag im letzten Schritt, wird jeweils der Wert für *C_HLEVEL* um eins von 1 ausgehend erhöht und abschließend ebenfalls gespeichert.
- Der Name und die Anzahl an passenden Patienten werden direkt in die entsprechenden i2b2-Felder übernommen.
- Da der *C_BASECODE* einen i2b2-Eintrag eindeutig identifizieren kann, werden in diesem eine Kombination aus *ExtID* und *Project* gespeichert, da diese wiederum PaDaWaN-Einträge eindeutig identifizieren. Zur Unterscheidung von nativen Basecodes wird außerdem noch „Padawan“ an den Anfang geschrieben. Diese drei Zeichenketten werden dann jeweils durch einen Doppelpunkt getrennt, da dieser auch von i2b2 als Trennzeichen innerhalb einzelner Basecodes verwendet wird.
- Wenn für *SingleChoice*-Einträge feste *Choices* oder für numerische Einträge Grenzwerte und Einheit vorhanden sind, so wird aus diesen das XML-Dokument zum Speichern in *C_METADATAXML* generiert.
- Damit die restlichen Informationen des *CatalogEntry* nicht verloren gehen, werden diese in eine Zeichenkette zusammengefasst und in *C_COMMENT* gespeichert.

– Je nachdem, ob der *CatalogEntry* noch untergeordnete Elemente hat oder nicht, wird der Eintrag entweder als „Folder“ oder als „Leaf“ gekennzeichnet, was dann die Darstellung in der Benutzeroberfläche festlegt.
– Als *SOURCESYSTEM_CD* wird der Wert „Padawan“ verwendet. Dies dient dann vor allem den Adaptern dazu, den Parameter *dw.index.i2b2.limitToPadawan* zu berücksichtigen und Anfragen entsprechend einzuschränken.
– Alle anderen Spalten haben jeweils fixe und zur Standardimplementierung von i2b2 passende Werte.

Eine Umwandlung in die andere Richtung, also von i2b2 nach PaDaWaN passiert mit den folgenden Schritten:

– Der Name, der Import-Zeitpunkt, die Anzahl an passenden Patienten und eventuell vorhandene fixe *Choices* beziehungsweise numerische Grenzwerte in Verbindung mit einer Einheit werden direkt in den entsprechenden PaDaWaN-Feldern gespeichert.
– *AttrID* und *ParentID* werden hier jeweils zunächst auf -1 gesetzt, da deren eigentlichen Werte dann erst vom *I2B2CatalogAdapter* bestimmt werden.
– Der Datentyp wird in Abhängigkeit von *C_COLUMNDATATYPE* entweder auf *Number* oder *Text* gesetzt. Falls *Choices* vorhanden sind, wird der Datentyp *SingleChoice* verwendet.
– Als *Project* wird der erste Teil und als *ExtID* der Rest des Pfades jeweils ohne die Backslash-Zeichen am Anfang verwendet.
– Alle restlichen Felder werden in einen String kombiniert und als *Description* gespeichert.

Neben dem beschriebenen *CatalogAdapter* verwendet der CatalogManager noch einen *CatalogChoiceAdapter*, einen *CatalogNumDataAdapter* und einen *CatalogCountAdapter*. Diese sind im PaDaWaN getrennt vom *CatalogAdapter*, da dort *Choices*, zusätzliche Daten zu numerischen Katalogeinträgen und die Anzahl an zu den Einträgen passenden Elementen in eigenen Tabellen stehen. Da dies alles in i2b2 in einer einzelnen Tabelle zusammengefasst ist, wurden für diese Adapter nur die Methoden zum Löschen und Hinzufügen neuer Daten implementiert. Diese funktionieren jeweils so, dass sie den zugehörigen *CatalogEntry* entsprechend anpassen und anschließend die Methode des *CatalogAdapter* zum Aktualisieren eines Eintrags verwenden. Die Methoden um die passenden Anzahlen an Fällen und Informationen neu zu berechnen werden vom *I2B2CatalogNumDataAdapter* nicht unterstützt, da diese Funktionen und Werte in i2b2 nicht vorgesehen sind. Zum Berechnen der Anzahl passender Patienten gibt es eine Methode der ONT-XML-Schnittstelle. Der entsprechende Aufruf ist jedoch auskommentiert, da dieser in der aktuellen Version von i2b2 selbst direkt über die i2b2 Workbench auf einem unveränderten i2b2-Demodatensatz ausgeführt, fehlschlägt. Sollte dies in einer zukünftigen Version von i2b2 wieder funktionieren, kann die Auskommentierung der entsprechenden Zeilen einfach entfernt werden. In i2b2 wird diese Zahl lediglich verwendet, um sie in den Oberflächen neben den einzelnen Konzepten anzuzeigen, weswegen durch ein Fehlen dieses Wertes keine wirklichen Einschränkungen in der Funktionalität von i2b2 entstehen.

Zur besseren und vor allem auch schnelleren Verwendung mit i2b2 wurde der *I2B2CatalogManager* um noch einige zusätzliche Funktionen verglichen mit dem PaDaWaN-CatalogManager erweitert:

- Beim Laden des Kataloges wird die Zuordnung der *AttrIDs* sowohl zu den einzelnen *C_BASECODE*-Werten als auch zu den einzelnen Pfaden gespeichert. Damit können diese dann jeweils mit einer Anfrage an den *I2B2CatalogManager* direkt ineinander umgewandelt werden, ohne dass jedes Mal im Hintergrund eine Datenbankabfrage nötig ist.
- Um native i2b2-Einträge ihren Elterneinträgen zuordnen zu können, wurde außerdem eine Methode ergänzt, um den *CatalogEntry* eines Elterneintrags anhand des Pfads des Kindereintrags abfragen zu können. Diese Methode behebt auch die Probleme, die durch Fehler im i2b2-Demodatensatz beim Einlesen dieses Kataloges sonst auftreten würden.

Viele der oben beschriebenen Funktionen sind auch direkt mit der XML-Schnittstelle der ONT-Komponente von i2b2 möglich. Die damit umsetzbaren Funktionen wurden im Rahmen dieser Arbeit auch für diese Schnittstelle implementiert und können mithilfe der *I2B2OntologyRequest*-Klasse verwendet werden. Da diese Schnittstelle jedoch deutlich langsamer als ein direkter Zugriff auf die Datenbank ist, wird diese Möglichkeit aktuell in keiner der anderen Klassen genutzt, weswegen sie auch im Rahmen dieser Arbeit nicht genauer beschrieben wird. Ein Beispiel für den wirklich deutlichen Geschwindigkeitsunterschied ist das Laden aller Einträge des i2b2-Demodatensatzes in den Arbeitsspeicher: Dies dauert direkt aus der Datenbank einige Sekunden und über die XML-Schnittstelle mehrere Stunden.

5.1.2 I2B2InfoManager

Auch der InfoManager funktioniert mithilfe von Adaptern. Außerdem benötigt er eine Instanz eines CatalogManager, welche im Falle des *I2B2InfoManager* ein *I2B2CatalogManager* sein muss.
Der wichtigste Adapter ist dabei der *InfoAdapter*. Dieser unterstützt verschiedenste Anfragen zum Abfragen von Informationen mit gewissen Einschränkungen oder auch aller Informationen. Auch können verschiedene Anzahlen oder nur die Werte bestimmter Informationen abgefragt werden. Der *I2B2InfoAdapter* unterstützt dabei sämtliche Funktionen, die für die Funktionalität des InfoManager benötigt werden. Da die meisten Funktionen in dem Adapter sehr ähnlich zu ihren PaDaWaN-Gegenstücken funktionieren, werden im folgenden hauptsächlich die Aspekte beschrieben, in denen sich die Adapter unterscheiden:

- Auch dieser Adapter legt bei seiner Initialisierung als erstes eine Datenbankverbindung an. Dazu werden die folgenden Konfigurationsparameter genutzt:
  - *dw.index.i2b2.db.type*, *dw.index.i2b2.db.address* und *dw.index.i2b2.db.port* wurden bereits oben für den *I2B2CatalogAdapter* beschrieben und werden auch hier mit identischer Funktionalität genutzt.
  - Da in i2b2 die Fakten in der Regel in einer anderen Datenbank als die Katalogeinträge stehen und diese auch einen anderen Login verwendet, lässt

sich der Datenbankzugriff unabhängig von den entsprechenden Katalog-Parametern mithilfe von *dw.index.i2b2.db.database* für den Namen der Datenbank sowie mit *dw.index.i2b2.db.user* und *dw.index.i2b2.db.password* für die Anmeldung an dieser Datenbank festlegen.

Solche Konfigurationsparameter stehen im PaDaWaN generell in einer speziellen Konfigurationsdatei aus jeweils einem Parameter mit zugehörigem Wert pro Zeile. Diese Parameter werden auch verwendet, um unter anderem die nötigen Werte zum Zugriff auf die PaDaWaN-Datenbank sowie auf die anderen Engines anzugeben.

– Wenn eine Zeile der Datenbank von einer der Anfragen zurückgegeben wurde, so wird diese wie folgt in eine PaDaWaN-Information umgewandelt, da eine solche von vielen Methoden des *InfoAdapter* als Rückgabewert erwartet wird:
  - Die Werte für die Kennung des Patienten und des Falls, sowie der Mess- und Importierungszeiptunkt werden direkt in die entsprechenden Informationsfelder übernommen.
  - Die *CONCEPT_CD* wird mithilfe des *I2B2CatalogManager* in die zugehörige *AttrID* umgewandelt.
  - Wenn der aktuell eingelesene Datensatz aus dem PaDaWaN exportiert wurde, so wird der Wert der *PROVIDER_ID* als *InfoID* und der der *INSTANCE_NUM* als *Ref* übernommen. Die genaue Umwandlung von PaDaWaN-Informationen in das i2b2-Format wird weiter unten noch beschrieben.
  - Falls sich um einen nativen i2b2-Fakt handelt, so erhält dieser eine *Ref* von 0 und eine *InfoID* die um eins über der zuletzt vergebenen liegt. Hierbei wird auch sichergestellt, dass diese nicht bereits von einer PaDaWaN-Information, die in i2b2 gespeichert wurde, verwendet wird.
  - Als nächstes werden in Abhängigkeit des Wertes der *VALTYPE_CD* *ValueDec*, *Value* und *ValueShort* aus den entsprechenden i2b2-Feldern ausgelesen.
– Bei einem Import neuer Daten müssen bei i2b2 nicht nur die Faktentabelle, sondern auch die Mapping- und Dimensionstabellen befüllt werden. Zu diesem Zweck wird eine neue Instanz der *I2B2PatientDataObject*-Klasse erzeugt, der wiederum die neuen Informationen übergeben werden. Diese generiert ein i2b2-PDO-Objekt im XML-Format aus den folgenden Bestandteilen:
  - Für die Mapping-Tabellen werden die Patienten beziehungsweise Fallkennungen in einem *pidSet* beziehungsweise *eidSet* zusammen mit der Zeichenkette *Padawan* als Bezeichner des Systems, aus dem die Kennungen stammen, gespeichert.
  - Im *ConceptSet* werden die Pfade aller von den übergebenen Informationen verwendeten Katalogeinträge zusammen mit deren Basecodes und Namen gespeichert.
  - Das *ObservationSet* enthält dann die eigentlichen Fakten. Dazu werden die Patienten- und Fallkennung sowie der Messzeitpunkt direkt übernommen.

Die *InfoID* wird als *PROVIDER_ID* und der Wert von *Ref* als *INSTANCE_NUM* gespeichert.
Der Wert der *AttrID* wird in den zugehörigen *C_BASECODE* umgewandelt.
Bei einem Datentyp von *Number* wird *VALTYPE_CD* auf „N“ gesetzt und der Wert in *NVAL_NUM* gespeichert. Sind numerische Grenzwerte für den Katalogeintrag vorhanden, so wird außerdem bestimmt, ob der aktuelle Wert zu hoch oder zu niedrig ist und die *VALUEFLAG_CD* entsprechend gesetzt. Auch bei einem Datentyp von *DateTime* wird diese *VALTYPE_CD* in Verbindung mit dem Zeitpunkt in Millisekunden als Wert gespeichert.
Dem Datentyp *Text* wird der *VALTYPE_CD* „B“ zugeordnet und der Wert als *OBSERVATION_BLOB* abgelegt.
Für alle anderen Datentypen wird als *VALTYPE_CD* „T“ in Verbindung mit dem eigentlichen Wert in *TVAL_CHAR* genutzt.

- Die Dimensionstabellen können neben den jeweiligen Kennungen noch beliebige andere Daten zu den einzelnen Patienten beziehungsweise Fällen enthalten. Um diese Daten anhand der PaDaWaN-Informationen setzen zu können, können die dafür vorgesehenen Spalten einzelnen PaDaWaN-Katalogeinträgen zugeordnet werden. Dafür muss eine Datei mithilfe des *dw.index.i2b2.fixedPatientDataMappingFile*-Konfigurationsparameters angegeben werden, die diese Zuordnung enthält. Die Zeilen in ihr, müssen wie folgt strukturiert sein: „*<PaDaWaN-ExtID>#<PaDaWaN-Project>#<'patient' oder 'event' (je nachdem zu welcher Dimensionstabelle die aktuelle Zeile gehört)>#<Name der Spalte in der Dimensionstabelle>#<Datentyp dieser Spalte (zum Beispiel 'int' oder 'string')>*“. Wird dann eine zu den dort angegebenen Einträgen gehörende Information übergeben, werden die Spalten entsprechend befüllt. Die Inhalte der Dimensionstabellen werden dann als *patientSet* und *eventSet* zusammengefasst.

Wenn die Anzahl an Patienten im *I2B2PatientDataObject* einen bestimmten Grenzwert überschreitet, oder sobald die *commit*-Methode des *InfoAdapter* aufgerufen wird, werden die neuen Werte in die Datenbank übernommen. Da eine Patienten- beziehungsweise Fallkennung aus dem PaDaWaN in i2b2 eventuell bereits vorhanden sein kann, wird dabei jeweils zunächst die nächste freie i2b2-Kennung ermittelt und deren Zuordnung zu der jeweiligen PaDaWaN-Kennung in den Mappingtabellen gespeichert. Dies umgeht auch das Problem, dass dadurch entsteht, dass die numerischen Kennungen des PaDaWaN länger sein können, als es für die numerischen Kennungen in der Standardimplementierung von i2b2 maximal möglich ist. Die Daten für die restlichen Tabellen verwenden dann die so generierten i2b2-Kennungen. Um diese Zuordnung bei Methoden des *InfoAdapter*, die eine *PID* beziehungsweise *CaseID* aus einem PaDaWaN-System verwenden wieder abfragen zu können, sodass dann für die eigentliche Anfrage die i2b2-Kennung benutzt werden kann, wurde der *I2B2InfoAdapter* um entsprechende Methoden erweitert.

Falls sich die variablen Spalten der Patienten- und Fall-Dimensionstabellen von denen der i2b2 Standardimplementierung unterscheiden, so müssen die Namen dieser Spalten in einer separaten Datei pro Tabelle stehen. Diese Dateien müssen dann pro Zeile genau einen Spaltennamen enthalten. Der Pfad zu diesen Dateien wird anschließend mithilfe der Konfigurationsparameter *dw.index.i2b2.patientDimensionDataColumnsFile* und *dw.index.i2b2.encounterDimensionDataColumnsFile* angegeben werden.
Falls nach einem abgeschlossenen Importvorgang noch weitere Daten importiert werden sollen, wird mit diesen wieder eine neue Instanz der *I2B2PatientDataObject*-Klasse erzeugt.

- Sämtliche aus dem PaDaWaN importierte Daten erhalten die *SOURCESYSTEM_CD* „Padawan“. Damit ist es bei einem Wert von „true“ des Konfigurationsparameters *dw.index.i2b2.limitToPadawan* möglich, alle Anfragen des *I2B2InfoAdapter* auf aus dem PaDaWaN stammende Daten zu beschränken. Diese Beschränkung wird außerdem immer gemacht, wenn eine Methode des Adapters benutzt wird, die die *InfoID* als Parameter erhält, da diese ausschließlich bei PaDaWaN-Einträgen vorkommt.
- Zusätzlich wurde der Adapter um jeweils eine Methode ergänzt, die alle Informationen zu einer Patienten- beziehungsweise zu einer Fallkennung aus sämtlichen Tabellen löscht.

Eine erzeugte Instanz der *I2B2PatientDataObject*-Klasse kann auch in ihrer XML-Form in eine Datei exportiert werden, damit diese Datei anschließend der XML-Schnittstelle der CRC-Komponente zum Import von Daten übergeben werden kann. Dies ist mithilfe der Klassen *I2B2QueryToolRequest* und *I2B2PublishDataRequest* möglich. Da dies jedoch wieder deutlich langsamer als ein direkter Zugriff auf die Datenbank ist, werden diese Klassen hier nicht genauer beschrieben.
Als zweiten Adapter benötigt der *InfoManager* einen *DeleteAdapter*. Dieser speichert Informationen darüber, welche Daten aus dem System zu welchem Zeitpunkt gelöscht wurden. Da diese Funktion von i2b2 direkt nicht unterstützt wird, legt der *I2B2DeleteAdapter* eine eigene Tabelle dafür in der CRC-Datenbank mit derselben Struktur wie die entsprechende PaDaWaN-Tabelle an und nutzt diese anschließend auch identisch zum entsprechenden PaDaWaN-Adapter. Für die Methoden, die mehrere Informationen als gelöscht markieren, werden die dazu benötigten Informationen zunächst mithilfe des *I2B2InfoAdapter* abgefragt.

5.1.3 I2B2IndexCreator

Zum Transfer von PaDaWaN-Daten in ein anderes System bietet der PaDaWaN die Klasse *AbstractDataSource2Index*. Diese wurde zum vereinfachten Transfer von Daten eines PaDaWaN-Systems in i2b2 mithilfe der Klasse *I2B2IndexCreator* implementiert.
Falls der i2b2-Index komplett neu aufgebaut werden soll, werden zunächst alle Fakten mithilfe des *I2B2InfoAdapter* gelöscht. Der *I2B2IndexCreator* erzeugt dann zunächst anhand der übergebenen Informationen eine Instanz der *I2B2PatientDataObject*-Klasse. Soll der i2b2-Index nicht neu aufgebaut werden, sondern

werden nur neue und veränderte Daten an den *I2B2IndexCreator* übergeben, so wird vor dem Speichern der neuen Einträge eines Patienten jeweils alle dessen bisher gespeicherte Daten mithilfe des *I2B2InfoAdapter* gelöscht. Nach einem bestimmten Grenzwert an Patienten im *I2B2PatientDataObject* und auch nachdem sämtliche vorhandene Informationen an den *I2B2IndexCreator* übergeben wurden, wird das *I2B2PatientDataObject* zum eigentlichen Import dem *I2B2InfoAdapter* übergeben und für weitere Daten anschließend ein neues *I2B2-PatientDataObject* erzeugt.
Nachdem dann alle Informationen übertragen wurden, werden alle Einträge des PaDaWaN-CatalogManager in eine CSV-Datei exportiert, welche anschließend vom *I2B2CatalogManager* nach einem Löschen seiner bisherigen Einträge wieder importiert wird. Dadurch wird dann auch der komplette Katalog in i2b2 transferiert.

5.2 Nutzung der i2b2-GUI zur Abfrage des PaDaWaN

Neben dem Datentransfer zwischen i2b2 und PaDaWaN sollte im Rahmen dieser Arbeit eine Möglichkeit entstehen, die graphischen Benutzeroberfläche von i2b2 mit einem PaDaWaN-System im Hintergrund benutzen zu können.
Da diese Oberfläche ausschließlich die XML-Schnittstellen der einzelnen Komponenten von i2b2 verwenden, wurden diese Schnittstellen für den PaDaWaN nachgebaut. Dabei wurden jedoch nicht sämtliche Schnittstellen implementiert, sondern nur die, die für die grundlegenden Funktionen der Oberflächen erforderlich sind. Die Schnittstellen der IM-, WORK-, und FR-Zelle wurden dabei nicht berücksichtigt, da deren Funktionen im PaDaWaN nicht vorhanden sind.
Für die wichtigsten Funktionen der Oberflächen werden Anfragen der PM-, ONT-, und CRC-Komponente benötigt. Die dazu benötigten XML-Schnittstellen wurden mithilfe eines „Eclipse Dynamic Web Project“ auf Basis eines Tomcat-Servers in der Version 9.0.0.M17 umgesetzt. Dadurch lassen sich dann POST-Anfragen mit dem XML-Dokument der i2b2-Anfrage an die URL *http://<Server Adresse>:8080/I2B2Rest/<Name des Service>/<Name der Anfrage>* stellen, die wiederum mit einem XML-Dokument im i2b2-Format beantwortet werden. Die Konfigurationsdatei muss für diese Webschnittstellen unter „WebContent/WEB-INF/ressources/I2B2Test.props“ abgelegt werden. Ein Template dafür wird zusammen mit dem Code der Arbeit zur Verfügung gestellt.

5.2.1 PMService

Dieser Service stellt die benötigten Funktionen der PM-Komponente zur Verfügung. Diese wird direkt nach der Anmeldung an den Oberflächen als erste kontaktiert, um den Login zu überprüfen und anschließend die Konfiguration des Nutzers zu erhalten. Damit die im folgenden beschriebenen Funktionen mit den i2b2-Oberflächen genutzt werden können, müssen die Konfigurationsdateien der Oberflächen lediglich um die vollständige Adresse dieses Services ergänzt werden.
Mithilfe der *DWAuthenticator*-Klasse des PaDaWaN wird der eingegeben Benutzername zusammen mit dem Passwort verifiziert. Hierbei wird dem Nutzer

in der Oberfläche dann auch eine entsprechende Fehlermeldung angezeigt, falls die Daten nicht stimmen oder der *DWAuthenticator* nicht erreichbar ist.
Wenn die Anmeldung erfolgreich ist, wird die Konfiguration für den aktuellen Nutzer zurückgegeben. Neben vielen Standardparametern mit festen Werten enthält diese Konfiguration den vollständigen Namen des Nutzers, welcher in den Oberflächen angezeigt wird, und außerdem die Adressen für die beiden folgenden Services.
Die Weboberfläche blendet dann die WORK-Komponente automatisch aus, da hierfür keine Adresse übergeben wurde. In der Workbench hingegen werden in den Ansichten der nicht unterstützten Komponenten Fehlermeldungen angezeigt. Diese Komponenten können dort aber einfach geschlossen werden, was dann auch bei einer erneuten Anmeldung gespeichert bleibt.
Da es über die Oberflächen jeweils auch möglich ist, das Passwort das Passwort des aktuellen Nutzers zu ändern, wurde diese Funktion ebenfalls implementiert. Da dies allerdings vom *AuthManager* des PaDaWaN, der die Nutzerdaten verwaltet, nicht unterstützt wird, wird hierbei der aktuelle Nutzer zunächst gelöscht und anschließend mit dem neuen Passwort wieder angelegt.

5.2.2 OntologyService

Mit diesem Service wird der Zugriff auf den Katalog ermöglicht. Hierbei werden die folgenden Arten von Anfragen der ONT-Komponente unterstützt:

- Nach einem Anmelden an der Oberfläche werden immer zunächst die Katalogeinträge auf höchster Hierarchieebene abgefragt und angezeigt. Dazu werden sämtliche direkte Kinder des *Root-CatalogEntry* geladen. Die Zuordnung der Pfade zu den *AttrID*-Werten wird dabei im Arbeitsspeicher abgelegt, um diese bei den folgenden Abfragen nutzen zu können.
- Wenn ein Eintrag in der Hierarchie aufgeklappt wird, so werden dessen direkte Kindereinträge abgefragt. Hierzu wird dann der Pfad in der Anfrage mithilfe der gespeicherten Zuordnungen in die passende *AttrID* umgewandelt. Mit dieser werden dann wiederum die Kindereinträge aus dem PaDaWaN-CatalogManager geladen und angezeigt. Auch deren Pfad-zu-*AttrID*-Zuordnung wird im Arbeitsspeicher behalten. Bei dieser Anfrage kann außerdem ein Maximum festgelegt werden, bis zu dem die Kindereinträge nur angezeigt werden. Sofern dieses gesetzt ist und zu viele Kindereinträge für einen Elterneintrag vorhanden sind, wird eine entsprechende Fehlermeldung anstelle der eigentlichen Einträge zurückgegeben.
- Die Oberflächen erlauben außerdem das Durchsuchen aller Katalogeinträge sowie nur der Einträge unter einem bestimmten Eintrag auf höchster Hierarchieebene. Hierzu wird die Methode *getEntriesByWordFilter* des CatalogManager benutzt, welche den Suchbegriff als Parameter erhält und sämtliche Einträge, deren Name zum Suchbegriff passt, zurückgibt. In den Oberflächen kann hier außerdem noch festgelegt werden, ob der Suchbegriff nur im Namen enthalten, diesem exakt entsprechen, oder der Anfang beziehungsweise das Ende des Namens sein soll. Diese Information wird dann vom *OntologyService* auf die Namen der Suchbegriffe angewendet. Auch werden bei einer

Filterung auf einen bestimmten Eintrag auf höchster Hierarchieebene nicht passende Ergebnisse aus der Treffermenge entfernt. Abschließend wird noch überprüft, ob die Anzahl der verbleibenden passenden Einträge unter einem eventuell übergebenen Grenzwert liegt. Falls das so ist, werden die Einträge zurückgegeben und in den Oberflächen angezeigt.
- Außerdem kann noch nach konkreten Werten für den *C_BASECODE* gesucht werden. Wenn dieser in der im vorherigen Abschnitt beschriebenen Form eingegeben wird, wird er in *ExtID* und *Project* zerlegt und anhand dessen der zugehörige *CatalogEntry* gesucht und angezeigt.

Sämtliche Katalogeinträge müssen zur Anzeige in der Oberfläche in i2b2-Konzepte umgewandelt und in XML-Form ausgegeben werden. Dies wird mithilfe der oben bereits beschriebenen *I2B2OntologyEntry*-Klasse gemacht.

5.2.3 QueryToolService

Dieser Service gehört zur CRC-Komponente und ist für das Durchsuchen der medizinischen Daten zuständig. Für ihn werden die folgenden Anfragen unterstützt:

- Nach dem Laden der Oberfläche werden bei dieser Komponente als Erstes die Arten von unterstützten Rückgaben abgefragt. Diese werden dem Nutzer dann bei einer Suche zur Auswahl angeboten. Hierbei sind im Moment die reine Anzahl an passenden Patienten, sowie die gefundene Teilmenge an Patienten sowie Fällen möglich.
- Außerdem werden beim Starten die bisherigen Anfragen des angemeldeten Benutzers abgefragt. Da eine solche Speicherung von Anfragen in dem von i2b2 benötigten Format im PaDaWaN nicht vorgesehen ist, wird hierfür eine eigene Tabelle in der PaDaWaN-Datenbank angelegt. Diese enthält pro Zeile die Daten einer Instanz von *I2B2QueryResult*. Diese Daten sind die XML-Anfrage im i2b2-Format, der Zeitraum der Ausführung dieser Anfrage, der Name und die ID der Anfrage, der Beschreibungstext der Anfrageergebnisse und der Nutzer, der die Anfrage ausgeführt hat. Die Daten dieser Tabelle werden beim Anmelden an der Oberfläche für den aktuellen Nutzer in den Arbeitsspeicher geladen und anschließend die von der Anfrage an den *QueryToolService* festgelegte Zahl an letzten Anfragen zurückgegeben und angezeigt. Zu sehen ist dabei jeweils der Name der Anfrage.
- Die angezeigten Anfragen können aufgeklappt werden, wobei dann zunächst die zugehörigen Ausführungsinstanzen angezeigt werden. Dies ist im Fall dieses *QueryToolService* immer nur eine. Die Instanz lässt sich dann nochmal aufklappen, woraufhin die vom Nutzer für diese Anfrage gewählten Ergebnissinstanzen dargestellt werden. Die Anzahl an Patienten ist dabei direkt zu sehen. Die gefundenen Mengen an Patienten und Fällen lassen sich nochmal aufklappen, woraufhin deren Kennungen zu sehen sind. Diese werden mit einer separaten PDO-Anfrage abgerufen, die weiter unten noch erläutert wird. Alle anderen „Aufklappvorgänge“ sind ebenfalls jeweils separate Anfragen, die direkt Mithilfe der *I2B2QueryResult*-Instanz der gewählten Anfrage beantwortet werden.

- Die Namen der bisherigen Anfragen können außerdem durchsucht werden. Dabei ist es wieder möglich anzugeben, ob der Suchbegriff im Namen nur enthalten, diesem exakt entsprechen oder der Anfang beziehungsweise das Ende des Namens sein soll. Auch kann eine Sortierung angegeben werden, die festlegt, ob die Anfragen aufsteigend oder absteigend nach ihrem Ausführungszeitpunkt sortiert zurückgegeben werden sollen.
- Außerdem kann der Name von Anfragen und die Beschreibung von Ergebnisinstanzen verändert werden. Auch lassen sich einzelne Anfragen wieder löschen. All das wird dann immer auch direkt in der Datenbank entsprechend angepasst.
- Eine weitere Möglichkeit vergangener Anfragen ist, dass sie erneut in der Suchoberfläche angezeigt werden können. Dazu wird dann die im *I2B2QueryResult* gespeicherte XML-Suchanfrage zurückgegeben.
- Die wichtigste Funktion dieses Service ist aber das Ausführen neuer Suchanfragen. Diese werden in der Oberfläche zusammengebaut und in XML-Form an den *QueryToolService* übergeben. Dort wird dann zunächst eine neue Instanz der *I2B2QueryResult*-Klasse mit der Suchanfrage als Parameter angelegt. Diese wandelt, nachdem ihr eine noch nicht verwendete ID zugewiesen wurde, als erstes die i2b2-Anfrage in eine MXQL-Anfrage um. Je nach den vom Nutzer ausgewählten Rückgaben, ist dies entweder eine *onlyCount*-Anfrage, oder eine Anfrage, in der zwar nicht die konkreten Werte, aber die *PIDs* und je nach Auswahl auch die *CaseIDs* der gefundenen Ergebnisse abgefragt werden. Anschließend wird die so generierte Anfrage mit einer der PaDaWaN-Suchengines ausgeführt. Welche dabei verwendet werden soll, lässt sich mit dem Konfigurationsparameter *dw.index.i2b2.restEngine* festlegen. Mögliche Werte sind dabei „SQL“, „Solr“, „ES“ und „Neo4j“. Die gefundene Anzahl an Patienten wird zusammen mit dem Start- und Endzeitpunkt der Anfragenausführung im *I2B2QueryResult* gespeichert, welches abschließend in der Datenbank abgelegt wird. Die Ergebnisse werden dann in XML-Form an die Oberflächen zurückgegeben. Die gefundenen Mengen an Patienten- und/oder Fallkennungen werden zwar in der *I2B2QueryResult*-Instanz jedoch nicht mit in der Datenbank gespeichert, da dies je nach Anfrage auch sehr große Datenmengen sein könnten. Wenn nach einem Neustart der Oberfläche also diese IDs nochmal angezeigt werden sollen, so wird die zugehörige Anfrage erneut gestellt.
- Zusätzlich werden noch PDO-Anfragen unterstützt. Diese werden von i2b2 benutzt, um konkrete Daten abzufragen. Neben der Anzeige der Patientenbeziehungsweise Fallkennungen zu vorherigen Suchen, erlaubt dies die Benutzung von Plugins (wie zum Beispiel das *ExportXLS Analysis Tool*), mit denen auch die Werte von Fakten abgerufen werde können. Dazu wird in der Regel eine zuvor gefundene Menge von Patienten- oder Fallkennungen als Filtermenge genutzt. Diese werden dann mithilfe des zugehörigen *I2B2QueryResult* abgefragt und dem *IDFilter* einer MXQL-Anfrage übergeben. Die eigentliche Suche wird wieder mithilfe der Methoden in der *I2B2-QueryResult*-Klasse zusammengebaut, nur das dieses mal auch Werte zurückgegeben werden. Die Suchanfrage wird mit der eingestellten Engine aus-

geführt und die gefundenen Informationen mithilfe der weiter oben beschriebenen *I2B2PatientDataObject*-Klasse in ein i2b2-*ObservationSet* umgewandelt und in XML-Form zurückgegeben. Das genannte Export-Plugin zum Beispiel stellt die so erhaltenen Daten dann in tabellarischer Form dar.

Das Anlegen der sowie der Zugriff auf die Datenbank werden von der Klasse *I2B2ServiceStorage* übernommen. Diese verwaltet auch die geladenen und erzeugten *I2B2QueryResult*-Instanzen und speichert außerdem die weiter oben genannte Zuordnung von i2b2-Pfaden zu PaDaWaN-*AttrIDs*. Dies muss in einer separaten Klasse gemacht werden, da von den Service-Klassen selbst für jede einzelnen Anfrage automatisch eine neue Instanz erzeugt wird.

5.3 Nutzung des PaDaWaN zur Abfrage von i2b2

Neben dem, was bereits oben beschrieben wurde, sollte im Rahmen dieser Arbeit auch eine Option entstehen, um mithilfe des PaDaWaN direkt in i2b2 gespeicherte Daten abfragen zu können. Dafür wurden zwei verschiedene Möglichkeiten entwickelt:

5.3.1 I2B2AdapterFactories

Zunächst sollte es möglich sein, mit den PaDaWaN-Engines direkt Daten aus i2b2 indexieren zu können. Diese haben jeweils eine Klasse, die die *AbstractDataSource2Index*-Klasse implementiert, welche das Laden der neuen Daten übernimmt und diese an die Engines übergibt.
Die Klasse basiert hauptsächlich auf einem Catalog- und einem InfoManager. Da diese wie oben bereits beschrieben für i2b2 umgesetzt wurden, fehlten nur noch einige Adapter, die von dieser Klasse benötigt werden. Dies ist einerseits ein *IndexLogAdapter*, der für das Logging während der Indexierung verwendet wird und andererseits der *ParamsAdapter*, der beliebige Parameter mit einem zugehörigen Wert speichert. Dieser wird unter anderem dafür benutzt, um den Zeitpunkt zu speichern, bis zu dem bereits die vorhandenen Informationen indexiert wurden, um bei einer späteren Indexierung nur die neuen und veränderten Daten betrachten zu müssen. Da die Funktionen der Adapter von i2b2 nicht direkt unterstützt werden, wird für sie von den i2b2-Implementierungen dieser Adapter jeweils in der CRC-Datenbank eine eigene Tabelle angelegt, die dann komplett analog zu den entsprechenden PaDaWaN-Tabellen funktioniert.
Bei der Initialisierung einer *AbstractDataSource2Index*-Instanz werden eine *ClientAdapterFactory* und eine *QueryAdapterFactory* benötigt, um die Manager und Adapter zu erzeugen. Diese wurden somit ebenfalls für I2B2 implementiert. Hiervon können auch noch andere Adapter erzeugt werden, welche allerdings für die Verwendung von *AbstractDataSource2Index* nicht benötigt werden und deswegen auch nicht für i2b2 umgesetzt wurden.
Um nun einen bestehenden Indexierer mit Daten aus i2b2 verwenden zu können, müssen einfach nur die Klassen der *I2B2AdapterFactories* in der Konfigurationsdatei mit den Parametern *client.adapterFactoryClass* und *query.adapterFacto-*

ryClass ergänzt werden. Somit ließ sich zum Beispiel erfolgreich ein Elasticsearch-Index aus dem i2b2-Demodatensatz aufbauen.

5.3.2 I2B2GUIClient

Als alternative Möglichkeit zum Zugriff speziell auf i2b2-Daten, die dorthin aus einem PaDaWaN-System übertragen wurden, wurden für i2b2 außerdem ein *GUIQueryClient* zusammen mit einem *QueryRunner* und einem *QueryRunnable* gebaut.
Damit diese verwendet werden können, muss zunächst mithilfe der Konfigurationsparameter *dw.index.i2b2.server.address*, *dw.index.i2b2.server.port* und *dw.index.i2b2.server.path* die URL festgelegt werden, unter der die Schnittstellen der i2b2-Komponenten erreichbar sind. Über *dw.index.i2b2.server.user* und *dw.index.i2b2.server.password* muss außerdem der Login eines Nutzers der verwendeten i2b2-Instanz angegeben werden.
Wenn die obigen Klassen eine MXQL-Anfrage erhalten, so geben sie diese an die *I2B2Util*-Klasse weiter, die wiederum die *I2B2QueryParamsVisitor*-Klasse nutzt, um die einzelnen Elemente der Anfrage durchzugehen und sie an eine Instanz der Klasse *I2B2SearchRequest* zu übergeben. Diese wandelt die PaDaWaN-Umfrage dann in das XML-Format einer i2b2-Anfrage um. Dabei legt sie auch fest, ob es eine reine Count-Anfrage sein soll, oder ob auch die passenden Mengen an Patienten und/oder Fällen zurückgegeben werden sollen.
Dieser *I2B2SearchRequest* wird dann wiederum vom *I2B2Util* an die *I2B2QueryToolRequest*-Klasse übergeben, die die Anfrage ausführt und zunächst nur den gefunden Count zurückgibt. Eine *onlyCount*-Anfrage ist an dieser Stelle beendet und der Count wird vom *QueryRunnable* zur Ausgabe zurückgegeben.
Falls bei der Anfrage auch konkrete Werte abgefragt werden sollen, so wird mithilfe der gefundenen Menge an passenden Patienten oder Fällen ebenfalls mit der *I2B2QueryToolRequest*-Klasse und der *I2B2SearchRequest*-Instanz eine PDO-Anfrage verschickt.
Das so erhaltene XML-Dokument wird dann vom *I2B2Util* weiter verarbeitet, um zunächst die i2b2-Patienten- und Fallkennungen mithilfe der zurückgegebenen Mapping-Daten wieder in ihre entsprechenden PaDaWaN-*PIDs* beziehungsweise -*CaseIDs* umzuwandeln. Zusammen mit diesen Informationen werden dann aus den gefunden i2b2-Fakten analog zu dem weiter oben beschriebenen Verfahren PaDaWaN-Informationen erzeugt. Aus diesen wiederum lassen sich dann die benötigten *ResultCellData*-Objekte zusammenbauen, die dann vom *QueryRunnable* an das PaDaWaN-PostProcessing übergeben werden, welches die Ergebnisse abschließend in tabellarischer Form ausgibt.
Um den *I2B2GUIClient* zu testen, wurden die selben *Indexing-*, *QueryLogic-* und *Stornotests* die auch für die anderen PaDaWaN-Suchengines genutzt werden, verwendet. Sämtliche der dort aktuell unterstützen Tests lassen sich auch erfolgreich auf einem in i2b2 importierten PaDaWaN-Demodatensatz ausführen.
Der entwickelte *I2B2GUIClient* kann dadurch, dass er die entsprechenden PaDaWaN-Klassen implementiert auch direkt in die PaDaWaN-Oberflächen eingebun-

den werden, um damit dann in ein i2b2-System übertragene PaDaWaN-Daten abfragen zu können.

6 Evaluation der Geschwindigkeiten beider Systeme

Abschließend sollten die beiden Systemen noch hinsichtlich ihrer Geschwindigkeit miteinander verglichen werden. Dazu wurden zufällige Datensätze generiert, auf denen dann wiederum zufällige Anfragen mit beiden Systemen ausgeführt wurden. Sowohl zur Daten- als auch zur Anfragengenerierung wurden bestehende Klassen[7] verwendet. Der genaue Ablauf der Tests wird im folgenden beschrieben:

6.1 Testdatensätze

Da das Ziel der Geschwindigkeitstests vor allem war, die Skalierung der Systeme bei unterschiedlich großen Datenmengen zu betrachten, wurden drei verschiedene Datensätze generiert. Für jeden davon wurde als erstes ein Katalog aus insgesamt 10.000 Einträgen erzeugt. Jeweils 100 dieser Einträge wurden dabei unter einem Eintrag auf höchster Hierarchieebene zusammengefasst. Der Datentyp der Einträge wurde zufällig generiert, sodass mit einer Wahrscheinlichkeit von 50% ein numerischer, zu 40% ein boolescher und zu 10% ein Eintrag vom Datentyp *Text* entsteht. Zu jeweils einem so erzeugten Katalog wurden dann Daten für Patienten generiert. Hierbei wurde ein Datensatz mit 250.000, einer mit 500.000 sowie einer mit 1.000.000 Patienten erzeugt. Dabei wurden wiederum pro Patient 3 Fälle, pro Fall 5 Dokumente und pro Dokument 10 Fakten erzeugt. Jeder einzelne Patient erhält somit 150 Fakten, was in den Datensätzen eine Gesamtzahl von 37,5 beziehungsweise 75 sowie 150 Millionen Fakten ergibt. Für jeden einzelnen Fakt wurde zunächst einer der generierten Katalogeinträge zufällig ausgewählt. Der Messzeitpunkt wiederum wird ebenfalls zufällig im Zeitraum vom 01.01.2000 (inklusiv) bis zum 01.01.2016 (exklusiv) gewählt. Dabei wird außerdem darauf geachtet, dass die Fakten nur innerhalb eines Zeitraums von 20 Tagen liegen. Bei booleschen Katalogeinträgen, wird der Fakt direkt erzeugt. Für numerische Einträge wird ein zufälliger Wert zwischen 0 und 100 mit 2 Nachkommastellen generiert. Die Text-Werte wiederum werden aus 1000 mit Leerzeichen verknüpften Nummern aus dem Bereich von 100 bis 999 erzeugt.
Sämtliche Tests wurden auf einem Rechner mit einer „Intel Core i7-4770"-CPU mit 3,40 GHz, 32 GB Arbeitsspeicher und ausschließlich SSD-Festplatten durchgeführt. Als Betriebssystem darauf installiert war „Windows Server 2012 R2 Standard".
Dieser Zufallsdatengenerator schreibt die generierten Daten direkt in eine PaDaWaN-Datenbank. Hierfür verwendet wurde ein „Microsoft SQL Server 2014", dem 16 GB Arbeitsspeicher zugeteilt wurden.
Als PaDaWan-Suchengine kam neben der SQL-Engine Solr in der Version 4.9.0

[7] Einsehbar nach entsprechender Freigabe unter https://gitlab2.informatik.uni-wuerzburg.de/misbased/misbased-sandbox/tree/develop/misbased-datagen

mit ebenfalls 16 GB zugewiesenem Arbeitsspeicher zum Einsatz. Mithilfe der bestehenden *TheDWIndexingApp*-Klasse wurden die erzeugten Datensätze in Solr indexiert.
Die einzelnen i2b2-Dienste wurden in einer virtuellen Maschine auf dem genannten Rechner ausgeführt. Auch dieser wurden 16 GB an Arbeitsspeicher zugewiesen. Als Betriebssystem kam hier Linux in der Variante „Ubuntu 14.04.5 LTS“ zum Einsatz, da so der oben beschriebene i2b2Wizard für Teile der Einrichtung verwendet werden konnte. Von i2b2 selbst wurde Version 1.7.07 benutzt. Als Datenbank kam auch hier der oben genannte MSSQL-Server zum Einsatz. Die generierten Daten wurden darin mit der im Rahmen dieser Arbeit entstandenen *I2B2IndexCreator*-Klasse übertragen.
Während der Datengenerierung und Indexerstellung wurde immer die Dauer sowie die resultierende Größe des jeweiligen Indexes gemessen. Die Ergebnisse dieser Messungen sind für den Datensatz aus 250.000 Patienten in Tabelle 1, für den aus 500.000 Patienten in Tabelle 2 und für den aus 1.000.000 Patienten in Tabelle 3 zu sehen. Alle angegeben Werte sind jeweils auf den nächsten ganzzahligen Betrag gerundet.

250.000 Patienten	PaDaWaN-SQL	Solr	i2b2
Zeit zur Indexerstellung	3 Std.	26 Std.	8 Std.
Größe des Index	114 GB	103 GB	128 GB

Tabelle 1. Bei der Datengenerierung und Indexerstellung gemessene Werte für den Testdatensatz aus 250.000 Patienten

500.000 Patienten	PaDaWaN-SQL	Solr	i2b2
Zeit zur Indexerstellung	5 Std.	48 Std.	14 Std.
Größe des Index	215 GB	192 GB	246 GB

Tabelle 2. Bei der Datengenerierung und Indexerstellung gemessene Werte für den Testdatensatz aus 500.000 Patienten

1.000.000 Patienten	PaDaWaN-SQL	Solr	i2b2
Zeit zur Indexerstellung	24 Std.	7 Tage	31 Std.
Größe des Index	396 GB	383 GB	473 GB

Tabelle 3. Bei der Datengenerierung und Indexerstellung gemessene Werte für den Testdatensatz aus 1.000.000 Patienten

Bei diesen Werten fällt auf, dass i2b2 - verursacht durch die komplexere Datenstruktur - am meisten Speicherplatz benötigt. Die PaDaWaN-Datenbank und der

Solr-Index sind hier jeweils sehr ähnlich von der Größe her. Da im PaDaWaN in der Regel sowohl die Datenbank als auch ein Solr-Index (und eventuell ein Index für weitere Suchengines) parallel zum Einsatz kommen, ist dessen Platzbedarf insgesamt höher, als der von i2b2.

6.2 Anfragegeschwindigkeit

Auf jedem der so erzeugten Datensätze, wurden anschließend verschieden komplexe Suchanfragen ausgeführt.
Dabei wurde immer nur die Anzahl passender Elemente und keine konkreten Werte abgefragt. Jede einzelne Art von Anfrage wurde 100 Mal pro Engine wiederholt und dabei jeweils die durchschnittliche Dauer der Anfragen zusammen mit der Standardabweichung ermittelt.
Damit die einzelnen Systeme jeweils die kompletten ihnen zugewiesenen Ressourcen uneingeschränkt nutzen konnten, wurden die Tests separat für jede Engine durchgeführt und die virtuelle Maschine für i2b2 und/oder der Solr-Server beendet, sofern sie für die aktuellen Anfragen nicht genutzt wurden. Auch wurde nachdem alle Tests für eine Engine durchgelaufen waren, immer der MSSQL-Server neu gestartet, damit dessen Zwischenspeicher bei keinem Durchgang noch Daten der vorherigen Durchgänge enthält.
Für die Anfragen wurde dann je nach gewünschtem Datentyp der Attribute zufällig ein jeweils dazu passender Katalogeintrag ausgewählt. Boolesche Attribute müssen dabei lediglich vorhanden sein, damit ein Patient als Treffer gewertet wird. Für numerische Attribute wird zufällig entschieden, ob gefundene Werte größer oder kleiner als ein zufälliger Wert aus dem oben beschriebenen Wertebereich sein müssen. Alternativ können diese Attribute auch so gesucht werden, dass sie nur vorhanden sein müssen. Bei der Suche innerhalb von Texten wurde ebenfalls eine zufällige Zahl aus dem oben erwähnten Bereich für Text-Katalogeinträge gewählt, welche dann mit dem *ContentOperator* „CONTAINS“ gesucht wurde.
Die folgenden Arten von Anfragen wurden zum Testen generiert:

- Ein einzelnes boolesches Attribut
- Ein einzelnes numerisches Attribut mit Einschränkung des Wertebereichs
- Ein einzelnes Text-Attribut
- Zwei mit *AND* verbundene boolesche Attribute
- Zwei mit *OR* verbundene boolesche Attribute
- Zwei boolesche Attribute, die im selben Fall vorhanden sein müssen
- Zwei numerische Attribute vom selben Katalogeintrag und ohne Einschränkung des Wertebereichs, von denen das Zweite mindestens einen Tag nach dem Ersten vorkommen muss
- Vier boolesche, vier numerische (mit Wertebereichseinschränkung) und vier Text-Attribute, die alle mit einem *OR* verbunden sind

Die dabei gemessenen Werte sind in Tabelle 4 für den Datensatz aus 250.000 Patienten, in Tabelle 5 für die 500.000 Patienten und in Tabelle 6 für den größten Datensatz mit 1.000.000 Patienten zu sehen. Zu beachten ist dabei, dass die

zeitlich-relative Suche von der PaDaWaN-SQL-Engine gar nicht und von Solr nur mithilfe einer Nachbearbeitung der Ergebnisse unterstützt wird.

250.000 Patienten	PaDaWaN-SQL	Solr	i2b2
1 - Bool	138 ± 71	57 ± 32	1160 ± 1231
1 - Num.	109 ± 83	58 ± 18	940 ± 442
1 - Text	718 ± 320	67 ± 59	3640 ± 494
2 - AND	68 ± 53	41 ± 5	1179 ± 387
2 - OR	283 ± 133	38 ± 8	1146 ± 344
2 - selber Fall	71 ± 17	35 ± 5	3167 ± 699
2 - relativ	-	6628 ± 2955	2017 ± 177
12 - OR	9764 ± 1507	224 ± 53	11956 ± 1453

Tabelle 4. Die durchschnittlichen Zeiten (in Millisekunden) der Ausführung von jeweils 100 Anfragen verschiedener Komplexität und deren Standardabweichungen über dem Datensatz aus 250.000 Patienten

500.000 Patienten	PaDaWaN-SQL	Solr	i2b2
1 - Bool	272 ± 112	53 ± 69	1100 ± 2027
1 - Num.	204 ± 114	72 ± 16	801 ± 486
1 - Text	986 ± 101	89 ± 70	3005 ± 848
2 - AND	126 ± 16	37 ± 7	1372 ± 726
2 - OR	525 ± 45	44 ± 6	1174 ± 220
2 - selber Fall	125 ± 9	36 ± 6	2502 ± 892
2 - relativ	-	20324 ± 4159	2372 ± 235
12 - OR	14656 ± 4607	510 ± 139	8950 ± 692

Tabelle 5. Die durchschnittlichen Zeiten (in Millisekunden) der Ausführung von jeweils 100 Anfragen verschiedener Komplexität und deren Standardabweichungen über dem Datensatz aus 500.000 Patienten

Insgesamt fällt bei den Tests auf, dass i2b2 in der Regel deutlich langsamer ist, als beide PaDaWaN-Engines und vor allem als Solr. Lediglich bei der relativen Suche ist i2b2 durch die hier für die Solr-Engine nötige Nachbearbeitung schneller. Dazu ist allerdings anzumerken, dass der PaDaWaN speziell für diese Art von Suchen die Neo4j-Engine unterstützt, mit welcher die relativen Suchen dann auch mit diesem System schneller möglich wären. Auch fällt auf, dass mit mehr Attributen pro Suche die Anfragezeiten aller Systeme merklich zunehmen. Ein weiterer deutlicher Unterschied zwischen den Systemen ist bei der Suche innerhalb von Texten zu beobachten. Dabei bestätigt sich, dass Solr speziell für diesen Zweck optimiert ist.
Im Hinblick auf die Skalierung der Systeme bei den unterschiedlich großen Da-

1.000.000 Pat.	PaDaWaN-SQL	Solr	i2b2
1 - Bool	402 ± 65	74 ± 31	1273 ± 532
1 - Num.	249 ± 104	96 ± 33	1099 ± 302
1 - Text	1010 ± 132	130 ± 66	9655 ± 1176
2 - AND	127 ± 13	57 ± 21	1426 ± 387
2 - OR	699 ± 158	118 ± 550	1573 ± 389
2 - selber Fall	122 ± 11	44 ± 8	3028 ± 490
2 - relativ	-	62236 ± 4368	3581 ± 423
12 - OR	18878 ± 1468	750 ± 111	40370 ± 3410

Tabelle 6. Die durchschnittlichen Zeiten (in Millisekunden) der Ausführung von jeweils 100 Anfragen verschiedener Komplexität und deren Standardabweichungen über dem Datensatz aus 1.000.000 Patienten

tensätzen kann beobachtet werden, dass die Anfragezeiten bei mehr vorhandenen Daten zwar bei allen Systeme tendenziell zunehmen, diese Zunahme aber mit Ausnahme der Anfrage aus 12 Attributen eher gering ausfällt.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurden sowohl der PaDaWaN als auch i2b2 als sehr umfangreiche Werkzeuge zur Verwaltung von und zum Zugriff auf medizinische Daten vorgestellt. Während vieles in beiden Systemen ähnlich funktioniert und umgesetzt ist, so gibt es auch eine Reihe von Unterschieden.
Der PaDaWaN verfügt über eine sehr simple Struktur und ist unter anderem dadurch beim Durchsuchen der Daten oft deutlich schneller. Auch die Möglichkeit zur Verwendung unterschiedlicher Suchengines trägt zu diesem Geschwindigkeitsvorteil bei. Außerdem ist die Anfragesprache des PaDaWaN wesentlich mächtiger als die von i2b2.
i2b2 hingegen hat eine sehr komplexe Datenstruktur, in welcher verschiedene Daten auch oft mehrfach abgelegt sind. Dies hat zwar einerseits den Vorteil, dass teilweise gerade mehr Metadaten als im PaDaWaN verwaltet werden können, bewirkt aber anderseits eine deutlich langsamere Anfragezeit. Ein weiterer Vorteil von i2b2 ist dessen modulare Struktur, durch welche sich einzelne Komponenten auf unterschiedliche Arten implementieren lassen, solange sie eine festgelegte Schnittstelle unterstützten. Außerdem ist es nur in i2b2 möglich, die Funktionalität der Oberflächen mit verschiedenen Plugins zu erweitern. Ein großer Nachteil dieses Systems ist hingegen, dass für die Abfrage einzelner Werte in der Regel zwei Suchanfragen nötig sind, während dies im PaDaWaN mit nur einer möglich ist.

7.1 Austauschbarkeit der Systeme

Verschiedene Möglichkeiten zur Verbindung der Systeme miteinander sind im Rahmen dieser Arbeit entstanden. So lassen sich nun unter anderem Daten beider Systeme in das jeweils Andere übertragen. Außerdem wurde eine Möglichkeit

geschaffen, die graphischen Benutzeroberflächen der Systeme mit jeweils dem anderen System im Hintergrund benutzen zu können.
Komplett austauschbar sind die Systeme dadurch allerdings noch nicht. Vor allem die Funktionalität der IM-, WORK- und FR-Komponenten fehlt im PaDaWaN komplett und müsste dort zunächst noch umgesetzt werden. Auch die unterschiedlichen Arten und Möglichkeiten der Nutzerverwaltung sind bei einer Verbindung der Systeme schwierig.
Folglich ist mit dieser Arbeit ein Anfang in Richtung der Austauschbarkeit von i2b2 und PaDaWaN miteinander gemacht worden, in dem diese für den Kern beider Systeme - also für die Kataloge und Fakten - realisiert wurde. Um diese jedoch vollständig zu ermöglichen, sind noch weitergehende Schritte notwendig.

Quellen

1. Oracle, *A Practical Guide to Clinical Data Warehousing* (aufgerufen am 29.03.2017, 15:00 Uhr),
   http://www.oracle.com/us/industries/life-sciences/guide-clinical-data-warehouse-ar-1563726.pdf
2. Dietrich, G. et. al., *Konzeption und Implementierung eines Data Warehouses für klinische Routinedaten an der Universitätsklinik Würzburg*,
   59. GMDS-Jahrestagung 2014 „Big Data und Forschungsinfrastruktur - Perspektiven für die Medizin“, 2014.
3. i2b2, *Offizielle Website* (aufgerufen am 15.02.2017, 16:00 Uhr),
   https://www.i2b2.org
4. Collen, M.F. and Ball, M.J., *The History of Medical Informatics in the United States*,
   Springer, London, 2015.
5. i2b2, *i2b2 Archived Source Code* (aufgerufen am 15.02.2017, 16:00 Uhr),
   https://www.i2b2.org/software/archive.html
6. i2b2 Community Wiki, *i2b2 Release Documentation* (aufgerufen am 15.02.2017, 16:00 Uhr),
   http://community.i2b2.org/wiki/display/RM/Release+Documentation
7. i2b2, *i2b2 Software* (aufgerufen am 16.02.2017, 13:00 Uhr),
   https://www.i2b2.org/software/index.html
8. Chueh, H. and Murphy, S., *The i2b2 Hive and the Clinical Research Chart*, 2006,
   https://www.i2b2.org/software/files/PDF/current/HiveIntroduction.pdf (aufgerufen am 16.02.2017, 13:00 Uhr)
9. i2b2, *i2b2 Cell Messaging - i2b2 Common Header* (aufgerufen am 16.02.2017, 13:00 Uhr),
   https://www.i2b2.org/software/files/PDF/current/MessageWrapper.pdf
10. i2b2, *i2b2.xsd* (aufgerufen am 16.02.2017, 13:00 Uhr),
    https://github.com/i2b2/i2b2-core-server/blob/master/edu.harvard.i2b2.xml/xsd/hive/msg_1.1/i2b2.xsd
11. HL7, *Message Header Segment* (aufgerufen am 16.02.2017, 13:00 Uhr),
    https://www.hl7.org/documentcenter/public/wg/conf/HL7MSH.htm
12. i2b2, *i2b2 Cell Messaging - ONT Cell* (aufgerufen am 18.02.2017, 18:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/Ontology_Messaging.pdf
13. i2b2, *i2b2 Cell Design - ONT Cell* (aufgerufen am 18.02.2017, 18:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/Ontology_Design.pdf
14. i2b2, *i2b2 Cell Messaging - CRC Cell* (aufgerufen am 19.02.2017, 15:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/CRC_Messaging.pdf
15. i2b2, *i2b2 Cell Design - CRC Cell* (aufgerufen am 19.02.2017, 15:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/CRC_Design.pdf
16. i2b2, *i2b2 Cell Messaging - PM Cell* (aufgerufen am 27.02.2017, 18:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/Project_Management_Messaging.pdf
17. i2b2, *i2b2 Cell Design - PM Cell* (aufgerufen am 27.02.2017, 18:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/Project_Management_Design.pdf
18. i2b2, *i2b2 Cell Messaging - WORK Cell* (aufgerufen am 28.02.2017, 14:00 Uhr),
    https://www.i2b2.org/software/files/PDF/current/Workplace_Messaging.pdf

19. i2b2, *i2b2 Cell Design - WORK Cell* (aufgerufen am 28.02.2017, 14:00 Uhr), https://www.i2b2.org/software/files/PDF/current/Workplace_Design.pdf
20. i2b2, *i2b2 Cell Messaging - IM Cell* (aufgerufen am 28.02.2017, 14:00 Uhr), https://www.i2b2.org/software/files/PDF/current/IM_Messaging.pdf
21. i2b2, *i2b2 Cell Design - IM Cell* (aufgerufen am 28.02.2017, 14:00 Uhr), https://www.i2b2.org/software/files/PDF/current/IM_Design.pdf
22. i2b2, *i2b2 Cell Messaging - FR Cell* (aufgerufen am 28.02.2017, 15:00 Uhr), https://www.i2b2.org/software/files/PDF/current/FR_Messaging.pdf
23. i2b2, *i2b2 Software Architecture - FR Cell* (aufgerufen am 28.02.2017, 15:00 Uhr), https://www.i2b2.org/software/files/PDF/current/FR_Architecture.pdf
24. i2b2, *i2b2 Cell Messaging - NLP Cell* (aufgerufen am 28.02.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/NLP_Messaging.pdf
25. i2b2, *i2b2 Software Architecture - NLP Cell* (aufgerufen am 28.03.2017, 15:00 Uhr), https://www.i2b2.org/software/files/PDF/current/NLP_Architecture.pdf
26. Deutsche Gesellschaft für Medizincontrolling e.V., *D002f Hauptdiagnose* (aufgerufen am 28.02.2017, 16:00 Uhr), http://foka.medizincontroller.de/index.php/DKR_D002f
27. GATE, *Offizielle Website* (aufgerufen am 28.03.2017, 15:00 Uhr), https://gate.ac.uk
28. UMLS, *Offizielle Website* (aufgerufen am 28.03.2017, 15:00 Uhr), https://uts.nlm.nih.gov/home.html
29. WEKA Data Mining, *Offizielle Website* (aufgerufen am 28.03.2017, 15:00 Uhr), http://www.cs.waikato.ac.nz/ml/weka/
30. i2b2, *i2b2 Cell Messaging - PFT Cell* (aufgerufen am 28.02.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/PFT_Messaging.pdf
31. i2b2, *i2b2 Functional Specification - Patient Count Plug-in* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/PatientCount_Functional_Specification.pdf
32. i2b2, *HPC Developer's Guide* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/HPC_Design.pdf
33. i2b2, *i2b2 User Guide - Text Analyzer View* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/TextAnalyzer_Help_Guide.pdf
34. i2b2, *Correlation Analysis Cell - A Tutorial* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/projects/correlation/Correlation_Analysis_Cell_Tutorial_10.pdf
35. i2b2, *i2b2 User Guide - Import Data View* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/Import_Help_Guide.pdf
36. i2b2, *i2b2 User Guide - Annotator Plug-in* (aufgerufen am 02.03.2017, 16:00 Uhr), https://www.i2b2.org/software/files/PDF/current/Annotator_Help_Guide.pdf
37. i2b2 Community Wiki, *i2b2 Wizard* (aufgerufen am 03.03.2017, 17:00 Uhr), http://community.i2b2.org/wiki/display/IDRT/200.+i2b2+Wizard
38. Solr, *Offizielle Website* (aufgerufen am 20.03.2017, 11:00 Uhr), http://lucene.apache.org/solr/
39. Lucene, *Offizielle Website* (aufgerufen am 20.03.2017, 11:00 Uhr), http://lucene.apache.org/
40. Elasticsearch, *Offizielle Website* (aufgerufen am 20.03.2017, 11:00 Uhr), https://www.elastic.co/de/products/elasticsearch

41. Neo4j, *Offizielle Website* (aufgerufen am 20.03.2017, 11:00 Uhr), https://neo4j.com
42. i2b2 Community Wiki, *IDRT Import and Mapping Tool* (aufgerufen am 23.03.2017, 16:00 Uhr), https://community.i2b2.org/wiki/pages/viewpage.action?pageId=9273415
43. i2b2, *i2b2 Software Architecture - CRC Cell* (aufgerufen am 24.03.2017, 13:00 Uhr), https://www.i2b2.org/software/files/PDF/current/CRC_Architecture.pdf
44. i2b2, *i2b2 Software Architecture - ONT Cell* (aufgerufen am 24.03.2017, 13:00 Uhr), https://www.i2b2.org/software/files/PDF/current/Ontology_Architecture.pdf

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde vorgelegt habe.

Würzburg, den 31. März 2017 _____

Leon Liman