ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Funktionsanalyse einer interdisziplinären Datenbank zu Optimierungszwecken

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Dominik Könke
dominik.koenke@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Rüdiger Pryss

2016

Fassung 17. August 2016

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Tinnitus ist ein weit verbreitetes, aber noch wenig erforschtes Krankheitsbild. Tinnitus eingehender zu erforschen und Daten über verschiedene Formen des Tinnitus zu sammeln, ist eines der Ziele der COST Action TINNET. Dabei fallen große Datenmengen an, die in einer webbasierten Datenbank gespeichert werden. Die Struktur dieser Datenbank soll überprüft und ihre Funktionalität, die sich momentan auf die grundlegenden Dinge beschränkt, erweitert werden.

Diese Arbeit analysiert die Struktur der Datenbank auf ihre Konsistenz hin. Es werden einige mögliche Änderungen erläutert. Ein in [1] vorgeschlagenes Validierungssystem wird umgesetzt. `FOREIGN KEY Constraints` ermöglichen eine effizientere Datenhaltung in der Datenbank. Mittels MySQL-Triggern werden E-Mail-Benachrichtigungen für Benutzerinnen und Benutzer des Systems ermöglicht, wenn Änderungen der Datensätze erfolgen.

Danksagung

Diese Arbeit wäre ohne tatkräftige Unterstützung von mehreren Seiten nicht möglich gewesen. Deshalb möchte ich an dieser Stelle folgenden Personen danken:

Meinem Betreuer Dr. Rüdiger Pryss für die hilfreichen Ratschläge und die zügige und klare Beantwortung aller Fragen.

Thai Chung, Johannes Delker und Gerald Könke für das Korrekturlesen dieser Arbeit und die hilfreichen Rückmeldungen.

Meinen Eltern Birgit und Gerald Könke, die mir immer den Rücken freihalten und mir auch und gerade in anstrengenden Zeiten, ganz besonders im Studium, zur Seite stehen und mich unterstützen.

Inhaltsverzeichnis

1

Einleitung

Tinnitus ist ein schwer greifbares, noch nicht ausreichend erforschtes Krankheitsbild. Viele mögliche Ursachen und unterschiedlich ausgeprägte Symptome sowie unterschiedliche Reaktionen verschiedener Patientinnen und Patienten auf verschiedene Behandlungen erschweren es zusätzlich, diese individuell wirksam zu behandeln. Die *Tinnitus Research Initiative* TRI (http://www.tinnitusresearch.org/) hat es sich zur Aufgabe gemacht, diese Hürden zu senken, Probleme zu mildern oder ganz zu beheben und neue Behandlungsmethoden zu finden. Sie sammelt im Rahmen des COST-Projekts TINNET Daten über Tinnitus-Patientinnen und -Patienten und deren Behandlung auf der ganzen Welt und versucht, mittels Fragebögen systematisch Grundlagen für neue Erkenntnisse zu schaffen. [2]

1.1 Problemstellung

Die bei der Auswertung der Fragebögen anfallenden Datenmengen sind nicht zu unterschätzen und werden deshalb strukturiert in einer webbasierten MySQL-Datenbank gespeichert. Diese soll im Rahmen dieser Arbeit in ihrer Funktionalität analysiert, erweitert und gegebenenfalls strukturell optimiert werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, die Anforderungen an die Datenbank des TRI zu analysieren und einen optimierten Entwurf bereitzustellen, der MySQL-Features nutzt und die Datenbank, wo nötig, strukturell erweitert und/oder optimiert.

1.3 Struktur der Arbeit

Nach der Bereitstellung einiger Hintergrundinformationen werden einige verwandte Arbeiten zur Orientierung diskutiert. Anschließend wird ein Anforderungskatalog mit funktionalen und nicht-funktionalen Anforderungen an die Datenbank erstellt, die die bereits bestehende Datenbank und die angestrebte Nutzung beschreiben. Diese Anforderungen werden im Hauptteil der Arbeit analysiert und in einem Datenbankentwurf möglichst vollständig umgesetzt. Anschließend wird die Analyse mit dem Anforderungskatalog abgeglichen, um die erreichten und auch die noch anderweitig umsetzbaren Optimierungen und Erweiterungen aufzuzeigen, bevor schließlich die Ergebnisse zusammengefasst werden und ein Ausblick auf weitere Schritte gegeben wird. Abbildung 1.1 zeigt die einzelnen Schritte in kompakter Form.

Abbildung 1.1: Ablauf der Arbeit

2

Hintergrundinformationen

Bevor die eigentliche Analyse durchgeführt werden kann, sind zum Verständnis der Strukturen im TRI zunächst einige Hintergrundinformationen nötig. Sie werden in diesem Abschnitt bereitgestellt.

2.1 COST

Die „European **Co**operation in **S**cience and **T**echnology“ – abgekürzt COST – wurde 1971 gegründet und ist das am längsten laufende europäische Rahmenprogramm zur Unterstützung internationaler Kooperationen zwischen Forschern, Ingenieuren und Studierenden in ganz Europa. Es ermöglicht die Zusammenarbeit von national geförderten Forschungsprojekten unterschiedlicher Disziplinen in sogenannten „Actions“. [3]

2.2 Tinnitus und TINNET

Tinnitus ist die Warhnehmung von Geräuschen ohne Vorhandensein eines akustischen Impulses von außen. In Europa haben über 70 Millionen Menschen Tinnitus, für 7 Millionen stellt es eine chronische Invalidität dar. Es gibt keine etablierten Behandlungsmethoden zur Heilung von Tinnitus, weswegen neue Behandlungen erforscht werden. Hirnforscher haben festgestellt, dass Tinnitus nicht etwa ein Problem im Ohr ist, sondern eine Folge veränderter neuronaler Aktivität im Gehirn. Trotz dieser Erkenntnis gestaltet sich die Entwicklung neuer Behandlungsmethoden aufgrund der unterschiedlichen Krankheitsbilder beziehungsweise Subtypen des Tinnitus schwierig. Das „**Tin**nitus Research **Net**work“ – abekürzt TINNET – ist eine COST Action, deren Finanzierung 2013 bewilligt wurde. Das Projekt unterstützt ein pan-europäisches Netzwerk. Dessen Hauptziele sind die Erleichterung

1. der Identifikation aussagekräftiger Kriterien zur Unterteilung von Tinnitus in Subtypen
2. der neurobiologischen Untermauerung dieser Typen und
3. ihrer Relevanz für das Ansprechen von Patienten auf Behandlungen

Das Projekt wird in mehreren Schritten durchgeführt, welche eine permanente koordinierte Zusammenarbeit von Wissenschaftlern, Technikern und Ärzten erfordern. [4] [5] [6]

Für das Design einer Website, die den Patientinnen/Patienten die Verwaltung ihrer Daten erleichtert [7] und ein Modul zur Analyse und grafischen Darstellung der Fragebogendaten [8] wurden bereits Konzepte vorgelegt.

2.3 Die Datenbank

Die Tinnitus Database (TDB) ist eine große und schnell wachsende internationale Patientendatenbank, an der sich jeder interessierte Arzt oder Forscher beteiligen kann. Die

Ärzte können schnell, umfassend und klinikübergreifend auf anonymisierte Patientendaten zugreifen. [2] [9]

Das ER-Diagram der aktuell verwendeten Datenbank ist in Anhang A zu finden.

In der Datenbank dient als Bezugseinheit eine Behandlungssitzung, gespeichert in der Tabelle `session`. Auf sie beziehen sich die Tabellen für die behandelten Patienten (referenziert über die Relationstabelle `patient_records`), die behandelnden Ärzte und die verwendeten Fragebögen zur Analyse. Letztere werden unterteilt in empirische Fragebogen, die jeweils in einer nach dem Fragebogen benannten Tabelle gespeichert werden, und nicht-empirische Fragebögen, welche in der Datenbank durch je eine Tabelle mit Meta-Informationen und eine Tabelle mit den eigentlichen Daten dargestellt sind.

3

Verwandte Arbeiten

In diesem Abschnitt werden Arbeiten besprochen, die sich mit ähnlichen Themen beziehungsweise Problemstellungen auseinandergesetzt haben wie diese.

3.1 Datenbanken im Einsatz

Das Buch „Datenbanken im Einsatz: Analyse, Modellbildung und Umsetzung“[10] beschreibt Arbeitsabläufe beim Erstellen von Datenbanken theoretisch und anhand von Beispielen und dient als Orientierung für das Vorgehen bei der Analyse in dieser Arbeit.

3.2 SEBIDA

SEBIDA, die **Se**x **Bi**as **Da**tabase (http://141.61.102.17/sebida/index.php), ist ein Projekt, das Daten über geschlechtsspezifisch verschieden in Erscheinung tretende Gene sammelt und analysiert.

Im Rahmen des Projektes wurde die Datenbank beschrieben. Eine vollständige Funktionsanalyse blieb jedoch aus.[11]

3.3 Wetter-Datenbank

Der Artikel beschreibt den strukturellen Aufbau einer Datenbank, die Daten von mehreren Wetterstationen sammelt und zusammenfasst. Dabei werden zunächst die anfallenden Daten funktional analysiert und vergleichbar gemacht, dann strukturiert und schließlich auch gleich wissenschaftlich analysiert. Ähnlich wie bei den Daten mehrerer Wetterstationen muss die TDB die verschiedenen ID-Formate für Patientinnen/Patienten der Behandlungszentren auf einen gemeinsamen Nenner bringen.[12]

3.4 Datenbank zu klinischen Nebenwirkungen

Fachgebietsverwandt wurde 2005 ein Artikel veröffentlicht, der die Struktur einer Datenbank zur Sammlung klinischer Nebenwirkungen in Medikamenten zur Verbesserung der Sicherheit von Patientinnen/Patienten sammelte. Im Artikel wird sowohl das Design des Frontends als auch die grundlegende Struktur der Datenbank beschrieben. Dabei werden die einzelnen Elemente der Datenbank wie in der aktuellen TDB in Objekte gruppiert. [13]

4

Anforderungskatalog

In diesem Kapitel wird ein Anforderungskatalog für den Datenbankentwurf erstellt, der im nächsten Kapitel analysiert und optimiert wird. Als Grundlage hierfür dienen die Weboberfläche der TDB, die von Robin Hagenlocher in seiner Masterarbeit entworfen wurde [1], das bereits erwähnte ER-Diagramm der Tinnitus Database in Anhang A – deren Bezeichner werden zum besseren Verständnis verwendet – und das Ziel, die Datenbank mittels Triggern zu erweitern. Es ist zu beachten, dass ein Systemadministrator sämtliche Aktionen durchführen kann. Daher wird er in Abschnitt 4.1 nicht explizit bei jeder Aktion als möglicher Aktor genannt. Außerdem werden die Anforderungen der Systemadministration gesondert in Abschnitt 4.2 aufgeführt. Die nicht-funktionalen Anforderungen werden in Abschnitt 4.3 behandelt.

4.1 Grundlegende Systemfunktionen

In diesem Abschnitt werden die Funktionen, die jede Benutzerin/jeder Benutzer im System verwenden kann, anhand der bestehenden Datenbank und der erwähnten Masterarbeit [1] katalogisiert.

Anforderung	Mögliche Aktoren	Benötigte /verwendete Daten	Beschreibung
Benutzerin /Benutzer im System registrieren	Jede Person mit Internetzugriff	Tabelle `user`	Eine Benutzerin/ein Benutzer wird der Datenbank hinzugefügt, üblicherweise über die Homepage-Registrierungs-oberfläche
Benutzerin /Benutzer aus dem System entfernen	Die betreffende Benutzerin/ der betreffende Benutzer	Eine Identifikation der Benutzerin/ des Benutzers, z.B. über ID oder Login-Credentials	Eine Benutzerin/ein Benutzer wird aus der Datenbank entfernt; hierbei wird abgefragt, ob eventuell vorhandene, von dieser Benutzerin/diesem Benutzer behandelte Patientinnen/ Patienten ebenfalls entfernt werden sollen. Diese werden ggf. mit `FOREIGN KEY Constraints` entfernt

Patientin/ Patient hinzufügen	Behandelnde Ärztin/ behandelnder Arzt oder Patientin/ Patient selbst	Tabelle `patients`	Die Patientin/der Patient wird in der Datenbank angelegt, üblicherweise durch die behandelnde Ärztin/den behandelnden Arzt. Diese/dieser wird auch als Erstellerin/Ersteller hinterlegt und in der user-Tabelle referenziert
Patientin/ Patient in `id_translation` eintragen	Behandelnde Ärztin/ behandelnder Arzt	Tabelle `id_translation`	Die Patientin/der Patient wird einem Zentrum zugewiesen, es wird die im System des Zentrums verwendete ID in `external_id` eingetragen und der `treatment_code` hinzugefügt; diese Aktion kann schon während oder direkt nach der Eintragung des Patienten/ der Patientin ins System erfolgen
Patientin/ Patient entfernen	Behandelnde Ärztin/ behandelnder Arzt	`patient_id`	Ein/e Patient/in wird aus der Datenbank entfernt. Dabei wird auf jeden Fall auch der Eintrag des Patienten in id_translation gelöscht. Es ist zu überprüfen, ob auch die erfassten Daten gelöscht werden sollen. Diese werden ggf. mit `FOREIGN KEY Constraints` entfernt
Behandlungssitzung erstellen	Behandelnde Ärztin/ behandelnder Arzt	Tabelle `session`	Eine neue Behandlungssitzung mit einem Patienten wird ins System eingetragen

Behandlungssitzung entfernen	Behandelnde Ärztin/ behandelnder Arzt	Referenz auf die Sitzung, etwa über Name und Patientin/Patient oder `session_id`	Die Daten einer Behandlungssitzung werden aus dem System entfernt
Weitere Behandlungsdaten eintragen	Behandelnde Ärztin/ behandelnder Arzt	Tabellen: `session_content_<Name>` wobei **<Name>** aus `adverse, comorbidity, concomittant, non_pharmalogical` sein muss	Weitere Behandlungsdaten werden ins System eingetragen; dabei müssen unterschiedliche Aspekte berücksichtigt werden, welche in unterschiedliche Tabellen eingetragen werden.
Medizinische Untersuchungsdaten eintragen	Behandelnde Ärztin/ behandelnder Arzt	Tabellen: • `otologic_examination` • `audiological_examination` • `patient_imaging_<Typ>`, wobei **<Typ>** aus `eeg, gen, meg, mrt` sein muss	Medizinische Untersuchungsdaten werden im System erfasst
Fragebogen ausfüllen	Behandelnde Ärztin/ behandelnder Arzt	`session_id`, Daten des Fragebogens; `session_content_id` und `session_content_name`, falls empirischer Fragebogen	Antworten auf einen Fragebogen werden im System erfasst

Statistiken abrufen	Behandelnde Ärztin/ behandelnder Arzt	Sämtliche erfassten medizinisch relevanten Daten der Datenbank	Behandelnde Ärztinnen/ Ärzte können zu Forschungs- und Vergleichszwecken etc. in einem Statistiktool Diagramme erstellen und dabei aus sämtlichen Daten auswählen und filtern
E-Mail-Benachrichtigungen	Behandelnde Ärztin/ behandelnder Arzt	Sämtliche erfassten medizinisch relevanten Daten der Datenbank	

Tabelle 4.1: Grundlegende Systemfunktionen

4.2 Systemadministration

In diesem Abschnitt wird der Anforderungskatalog um zusätzliche Funktionen für Administratorinnen und Administratoren erweitert, die Metadaten der Datenbank und die Funktionen in Abschnitt 5.5 der Masterarbeit [1] betreffen.

Anforderung	**benötigte/ verwendete Daten**	**Beschreibung**
Qualitäts-kontrolle	zu validierende Einträge bzw. deren IDs in der Datenbank	Administratorinnen/Administratoren können im System einzelne Behandlungssitzungen sowie empirische Fragebögen (in-)validieren
Fragebogen-verwaltung	Fragebogen-Tabellen	Administratorinnen/Administratoren können Fragebögen zur Datenbank hinzufügen, ändern und entfernen
Verwaltung der Behandlungs-zentren	Tabelle `centers`	Administratorinnen/Administratoren können Behandlungszentren zur Datenbank hinzufügen, ändern und entfernen
Benutzer-verwaltung	Tabelle `users`	Administratorinnen/Administratoren können Benutzerkonten erstellen, verändern und entfernen, insbesondere auch deren Rechte
Rollen-verwaltung	Tabellen `roles` und `possible_role_rights`	Administratorinnen/Administratoren können Benutzerrollen hinzufügen, entfernen und deren Rechte verändern

Metadaten-verwaltung	Tabellen: • `score_rules` • `possible_types` • `possible_sessions` • `possible_a` • `possible_adverse_events` • `possible_comorbidity_diseases` • `possible_concomittant_medications` • `possible_non_pharmalogical_interventions`	Administratorinnen/Administratoren können die Hintergrunddaten der Datenbank verwalten, um Vorgaben für bestimmte Inhalte abzubilden

Tabelle 4.2: Anforderungen der Systemadministration

4.3 Nicht-funktionale Anforderungen

In diesem Abschnitt werden nicht-funktionale Anforderungen, die die Datenbank direkt oder indirekt betreffen, erläutert.

Anforderung	**Beschreibung**
Datenbanksoftware	Die aktuelle Datenbank wurde mit MySQL implementiert; ein Wechsel ist nicht vorgesehen
Webanwendung	Damit alle Beteiligten weltweit das System nutzen, warten und/oder verwalten sowie Daten synchronisieren und darauf zugreifen können, ist das System webbasiert
Dokumentation	Der Aufbau der Datenbank wird mit Datenbankschemata beziehungsweise ER-Diagrammen dokumentiert
Echtzeitreaktion	Die Datenbank soll in der Lage sein, (nahezu) in Echtzeit, d.h. ohne deutliche Verzögerungen, auf Anfragen zu reagieren

Tabelle 4.3: nicht-funktionale Anforderungen

5

Funktionsanalyse

In diesem Kapitel wird die eigentliche Datenbankanalyse durchgeführt. Dies geschieht anhand der im vorigen Kapitel aufgestellten Anforderungen und des ER-Diagramms in Anhang A. In Abschnitt 5.1 wird die grundlegende Struktur beziehungsweise der Aufbau der Datenbank analysiert. In Abschnitt 5.2 werden die Tabellen für die Hintergrunddaten hinzugefügt, die von Administratorinnen und Administratoren verwaltet werden.

5.1 Grundlegende Struktur

In diesem Abschnitt wird die grundlegende Struktur der Datenbank analysiert; dabei werden Teile der Anforderungen aus den Abschnitten 4.1 und 4.2 mit dem bestehenden ER-Diagramm verglichen und gegebenenfalls ergänzt und daraus die Tabellen und Attribute abgeleitet. Die Zeitstempel, die in jeder Tabelle vorkommen, werden separat

in Abschnitt 5.1.6 besprochen. Die Löschung von Tabelleneinträgen, die mittels sogenannter `FOREIGN KEY`s in anderen Tabellen referenziert werden, wird in Abschnitt 5.3.2 besprochen. Die einzelnen Felder werden implizit als zwingend erforderlich, also mit dem Flag `NOT NULL` versehen, beschrieben. Nur wenn ein Feld auch fehlen kann, also optional ist, wird dies erwähnt. Die Typen der Attribute in der aktuellen Datenbank sind intuitiv klar und werden deshalb nicht explizit erwähnt. Sie sind im ER-Diagramm in Anhang A sichtbar.

5.1.1 Benutzerverwaltung

Um ein Benutzerkonto zu erstellen, sind bestimmte Daten erforderlich; zuallererst natürlich ein Login-Pseudonym und ein Passwort (im aktuellen System das Attribut `password`). Letzteres dient zur Authentifizierung der Benutzerin/des Benutzers, ersteres zur Identifikation. Diese kann zum Beispiel über eine E-Mail Adresse (`email`) oder einen selbstgewählten Benutzernamen (`username`) erfolgen; beides wird zur Registrierung im aktuellen System benötigt. Dieses speichert eine automatisch generierte `id` als Primärschlüssel sowie die Benutzerrolle (`role`) und die ID des Behandlungszentrums (`center_id`), zu dem die Benutzerin/der Benutzer gehört als Fremdschlüssel. Zusätzlich werden Vor- und Zuname des Benutzers (`firstname, lastname`) und die Sprache, in der die Benutzerin/der Benutzer das System verwendet (`lang`), gespeichert. Das im ER-Diagramm sichtbare optionale Attribut `remember_token` der Tabelle `users` ist eine für die Funktionalität der Tabelle als solche irrelevante Frameworkvariable. Diese Einträge gehören alle zusammen zu der Benutzerin/dem Benutzer, daher ist es sinnvoll, eine eigene Tabelle für Benutzer anzulegen und die Einträge als Attribute darin zu speichern, wie es auch im aktuellen System geschieht. Daher kann die Tabelle `users` 1:1 übernommen werden.

Die Änderung einzelner Attribute, zum Beispiel des Passworts, oder die Löschung des Benutzerkontos erfolgt über das System, das aus Eingabemasken für die Benutzerinnen/Benutzer entsprechende Datenbankanfragen generiert, oder durch Anfragen von Administratoren.

5.1.2 Patientenverwaltung

Ähnlich wie bei der Benutzerverwaltung ist auch eine Patientin/ein Patient am sinnvollsten als Entität mit Attributen zu betrachten. Dazu gehören die automatisch generierte ID der Patientin/des Patienten als Primärschlüssel (`patient_id`), als Fremdschlüssel die ID ihrer/seiner Patientengruppe und die Benutzer-ID der Erstellerin/des Erstellers des Patienteneintrags (`creator_id`, im Normalfall die/der behandelnde Ärztin/Arzt); das Geburtsdatum (`dateofbirth`) und das Geschlecht (`sex`). Für die Verwaltung der Daten der Patientin/des Patienten in Bezug auf das Behandlungszentrum gibt es im aktuellen System zusätzlich die Tabelle `id_translation`. Diese enthält die Attribute `translation_id` als Primärschlüssel, die `patient_id` der Patientin/des Patienten und die `center_id` des Behandlungszentrums als Fremdschlüssel sowie einen `treatment_code` für die externe Behandlungsdokumentation und die ID der Patientin/des Patienten im System des Behandlungszentrums (`external_patient_id`).

Diese Abtrennung ist sinnvoll, denn ein Patient kann in mehr als einem Behandlungszentrum gelistet sein. Deshalb ist die Beziehung der Tabellen `patients` und `id_translation` keine 1:1-Beziehung. Ansonsten könnte einfach die `patients`-Tabelle selbst die `center_id` als Fremdschlüssel zur direkten Referenz auf die `centers`-Tabelle verwenden, und alle genannten Informationen könnten in der `patients`-Tabelle gespeichert werden.

Die Patientengruppen werden in der Tabelle `patient_groups` gespeichert. Diese enthält eine automatisch generierte `patients_groups_id` als Primärschlüssel – hier wäre im Sinne der Konsistenz eine Umbenennung in `patient_groups_id` angebracht – den Namen der Patientengruppe in `group_name` sowie eine optionale Beschreibung in `group_info`. Die Auslagerung dieser Informationen in einer eigenen Tabelle erspart Duplikate der Informationen im Vergleich zu einer Speicherung in der `patients`-Tabelle, da mehrere Patienten in der selben Gruppe sein können, und ist deswegen sinnvoll.

Insgesamt gilt auch hier, dass die Struktur prinzipiell sinnvoll ist und übernommen werden kann.

5.1.3 Sitzungsverwaltung

Die Tabelle `sessions` ist im aktuellen System die zentrale Tabelle, das heißt alle anderen Tabellen, die direkt mit einer Sitzung zu tun haben – also die Metadaten der Sitzungstermine [1], die zugehörigen Patientinnen/Patienten der Sitzung (in der aktuellen Datenbank referenziert über die Tabelle `patient_records`, da eine Patientin/ein Patient mehrere Behandlungssitzungen durchlaufen kann), die Daten der empirischen Fragebögen[2] und die Metadaten der zusätzlichen Behandlungsinformationen[3], die Daten der audiologischen (`audiological_examniation`) und der otologischen Untersuchung (`otologic_examination`) – referenzieren sie über den automatisch generierten Primary Key `session_id` als zentrales Element der Datenbankstruktur. Man beachte, dass die Referenz der empirischen Fragebögen, der otologischen Untersuchung und der audiologischen Untersuchung momentan nicht im ER-Diagramm eingetragen ist, aber im aktiven System trotzdem besteht. Die medizinischen Untersuchungsdaten (siehe Abschnitt 5.1.4) referenzieren die Sitzung über die `patients`-Tabelle, da die Imaging-Daten einer Patientin/eines Patienten auch über mehrere Sitzungen hinweg verwendet werden können. Eine Sitzung referenziert ihrerseits die zugehörige Patientin/den zugehörigen Patienten über die Tabelle `patient_records` mittels des Fremdschlüssels `patient_record_id`. Jede Sitzung hat einen Namen (`session_name`) und einen Typen (`session_type`). Es wird mittels den Attributen `min` und `max` festgehalten, wie oft ein bestimmter `session_type` bei einer Patientin/einem Patienten vorkommen darf. Schließlich wird auch gespeichert, ob die Sitzung abgeschlossen (`inputstate_datafinish`), abgebrochen (`inputstate_dropout`) beziehungsweise nach Beendigung validiert (`inputstate_validated`) wurde. Es ist auch hier offensichtlich, dass die Einteilung sinnvollerweise beibehalten werden kann.

[1] Tabellen: `session_content_screening`, `session_content_baseline`, `session_content_wardround`, `session_content_final_wardround`, `session_content_catamnesis`

[2] Tabellen: `tschq` mit den angehängten Medikamenten in der Tabelle `tschq_medications`, `thi`, `bdi`, `mdi`, `cgi`, `tbf12`, `whoqol-bref`, `tinnitus_severity`, `tinnitus_questionnaire_gundh`

[3] Tabellen: `session_content_adverse` mit Referenz auf die eigentlichen Daten in `adverse`, analog `session_content_comorbidity` und `comorbidity`, `session_content_concomittant` und `concomittant` sowie `session_content_non_pharmalogical` und `pharmalogical`

Die bereits erwähnte Tabelle `patient_records` verknüpft die Behandlungssitzungen mit den Patientinnen/Patienten. Sie enthält eine automatisch generierte ID `patient_record_id` als Primärschlüssel und den Fremdschlüssel `patient_id` zur Verknüpfung mit der Tabelle `patients`. Außerdem listet sie die Sprache, in der die Patientin/der Patient behandelt wird (`language`). Dies ist schon allein deshalb notwendig, weil Patientinnen/Patienten nicht notwendigerweise ein Benutzerkonto haben und deshalb noch anderweitig bekannt werden muss, welche Sprache die Patientin/der Patient verwendet. Außerdem ermöglicht die Speicherung der Sprache in der Verknüpfungstabelle die Durchführung einer Behandlungssitzung in einer anderen Sprache als der, in der die Patientin/der Patient das System verwendet und die Durchführung verschiedener Behandlungssitzungen in unterschiedlichen Sprachen. Zum besseren Verständnis wäre auch hier unter Umständen eine kleine formale Änderung sinnvoll: Einerseits könnte man im Sinne der Konsistenz entweder das Attribut `language` in `lang` oder das Attribut `lang` der Tabelle `users` in `language` umbenennen. Andererseits haben die Attribute nicht die gleiche Bedeutung – das eine ist die Sprache, in der eine Benutzerin/ein Benutzer das System verwendet, das andere eben die Sprache, in der eine Behandlungssitzung durchgeführt wird. Beides ist also möglich. Falls für die Patientin/den Patienten bereits eine oder mehrere Behandlungssitzungen in der Datenbank hinterlegt sind, werden deren Typen im optionalen Attribut `existing_types` gespeichert. Dieser Eintrag ist auch der Hauptgrund, warum eine Auslagerung der genannten Daten in eine eigene Tabelle sinnvoll ist. Zwar hat jede Sitzung genau eine Patientin/einen Patientin, allerdings sind die Typen der bereits absolvierten Sitzungen relevant (bezüglich `min` und `max`, s. o.) und würden Eintragsduplikate in der `sessions`-Tabelle erzeugen. Außerdem hat die gesprochene Sprache nicht direkt mit den Untersuchungsdaten der Behandlungssitzung zu tun und kann deshalb sinnvollerweise ausgelagert werden. In der aktuellen Datenbank enthält die Tabelle zusätzlich ein Attribut `sessions_session_id`. Allerdings referenziert dieses keine andere Tabelle und ist auch nicht sinnvoll, da die Tabelle `sessions` bereits ihrerseits die Tabelle `patient_records` referenziert. Es kann also gestrichen werden. Mit dieser Änderung können auch die Tabellen `sessions` und `patient_records` übernommen werden.

5.1.4 Medizinische Untersuchungsdaten

Für die medizinischen Untersuchungen werden zwei Datensätze direkt in der Datenbank eingetragen: Die Ergebnisse der audiologischen Untersuchung (Untersuchung des Hörvermögens) in der Tabelle `audiological_examination` und die Ergebnisse der otologischen Untersuchung (Untersuchung der Ohren) in der Tabelle `otologic_examination`. Beide Tabellen haben eine automatisch generierte id (`audiological_examination_id` beziehungsweise `otologic_examination_id`) als Primärschlüssel, die `session_content_id` und den `session_content_name` (damit das System abfragen kann, bei welchem Termin die Untersuchung stattfand) als Verweis auf den zugehörigen Behandlungssitzungstermin sowie den `base_type` des zugehörigen Typs als Fremdschlüssel. Für die einzelnen Untersuchungsergebnisse steht jeweils ein weiteres Feld zur Verfügung. Da der Zustand des Ohrs sich wie der des ganzen Körpers beständig verändert und ältere Untersuchungsergebnisse eventuell nicht mehr relevant sind, können die Daten auch entfernt beziehungsweise überschrieben werden; hierfür enthält die Tabelle ein zusätzliche optionales Feld 'deleted_at' als Information, wann eine solche Aktion erfolgt ist. Beide Tabellen können ohne weiteres übernommen werden.

Die Tabellen für die Imaging-Daten [4] enthalten jeweils eine automatisch generierte ID als Primärschlüssel und die `patient_id` der zugehörigen Patientin/des zugehörigen Patienten als Fremdschlüssel. Da Imaging-Daten normalerweise in Bildform vorliegen, werden die Daten anderswo auf dem Server abgelegt und der zugehörige Pfad in einem Attribut `<Tabellenname>_path` (also zum Beispiel `patient_imaging_mrt_path`) angegeben. Optional kann eine Beschreibung der Datei im Attribut `<Tabellenname>_description` mit angegeben werden. Da auch Imaging-Daten veralten können, wird im Falle einer Löschung ein optionales Feld `deleted_at`, das ebenfalls alle vier Tabellen enthalten, gesetzt. Diese Einteilung listet die Daten sauber auf und kann deshalb problemlos übernommen werden.

[4] `patient_imaging_eeg`, `patient_imaging_gen`, `patient_imaging_meg`, `patient_imaging_mrt`

5.1.5 Fragebögen ausfüllen

Die nicht-empirischen Fragebögen `adverse`, `comorbidity`, `concomittant` und `non_pharmalogical` enthalten jeweils eine automatisch generierte ID als Primärschlüssel und die `session_content_id` der zugehörigen Metatabelle als Fremdschlüssel. Die einzelnen Fragen zu Behandlungen, Krankheiten und Nebenwirkungen werden jeweils in eigene Attribute eingetragen. Insgesamt sind die Tabellen intuitiv und können so beibehalten werden.

Analoges gilt für die empirischen Fragebögen, die den Sitzungstermin über die `session_content_id` referenzieren.

5.1.6 Zeitstempel

Das aktuelle System hat in jeder Tabelle zwei Zeitstempel vom Typ `DATETIME`, eines für die Erstellung des Eintrags (`created_at`) und einen für etwaige Veränderungen (`updated_at`). Diese werden jedoch datenbankintern nicht automatisch befüllt, obwohl dies Zeit und Arbeit sparen kann und leicht möglich ist: MySQL 5.6 stellt für Attribute des Typs `DATETIME` und `TIMESTAMP` Default-Werte bereit, die Erstellungs- und Aktualisierungs-/Veränderungszeitpunkt festhalten:
`CURRENT_TIMESTAMP` beziehungsweise
`CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP`.[14]
Zu beachten ist, dass die aktuell (August 2016) verwendete MySQL-Version des Servers, 5.5.50, es verbietet, dass mehr als ein TIMESTAMP die Systemvariable `CURRENT_TIMESTAMP` als `DEFAULT`- bzw. `ON UPDATE`-Parameter verwendet. Vor dem Einsatz dieser Werte wäre also ein Serverupdate notwendig.[15]

Verwendet man hierbei nun den Typ `DATETIME` - wie im aktuellen System - ergibt sich unter Umständen ein weiteres Problem: Die Zeiten werden als Systemzeiten abgespeichert. Da der Server eine eigene Systemzeit hat, spielt das momentan keine Rolle bezüglich der unterschiedlichen Zeitzonen der Anwender; bei einer Änderung der Serversystemzeit würden aber Unstimmigkeiten entstehen. Der Typ `TIMESTAMP` schafft hier

Abhilfe: Beim Speichern wird die Zeit in UTC umgewandelt und beim Auslesen wieder in Systemzeit zurückgerechnet.

5.2 Daten für die Administration

In diesem Abschnitt werden die noch nicht behandelten Anforderungen aus Abschnitt 4.2 besprochen, die von Administratorinnen/Administratoren verwaltet werden können. Die spezielle Benutzerverwaltung durch Administratorinnen/Administratoren beeinflusst die Datenbankstruktur nicht weiter, die Strukturen aus 5.1.1 reichen aus. Administratorinnen und Administratoren können im System über eine spezielle Oberfläche alle Benutzer verwalten (Abschnitt 5.5.2 in [1]).

5.2.1 Qualitätskontrolle

In der aktuellen Datenbank sind die Attribute für die in [1] vorgeschlagene Validierung der Fragebögen bezuglich eines Patienten (Abschnitt 5.5.1) beziehungsweise für einzelne Fragebögen als solche (Abschnitt 5.5.6) nicht vorhanden. Bei den einzelnen Fragebögen eines Patienten würde ein 'validated'-Attribut mit Wahrheitstyp (in MySQL `TINYINT(1)` beziehungsweise `BOOL` oder `BOOLEAN` als Typsynonym [16]) in den jeweiligen Fragebogen-Tabellen ausreichen. Etwas schwieriger ist die in Abschnitt 5.5.6 von [1] thematisierte Validierung der empirischen Fragebögen umzusetzen, da diese keine Metatabellen haben. Eine Möglichkeit wäre hier, ein Attribut `empiric_validation` des Typs `TINYINT(2)` anzulegen und Werte von 0-3 analog zu den vorgeschlagenen vier Validierungsstufen einzutragen. Eine Veränderung des Validierungsstatus wäre dann eine Änderung der gesamten entsprechenden Spalte der Fragebogentabelle. Die empirischen Fragebögen würden also sowohl das Attribut `validated` als auch das Attribut `empiric_validation` enthalten.

Diese Änderungen ermöglichen eine einfache Validierung der Fragebögen und werden deshalb im neuen ER-Diagramm in Anhang B umgesetzt.

5.2.2 Fragebogenverwaltung

Um Fragebögen hinzuzufügen, können Administratorinnen/Administratoren wie in 5.1.5 die Metaattribute und die einzelnen Fragen der Fragebögen in einer neuen Tabelle anlegen. Zur Veränderung werden die Attribute angepasst, und ein Fragebogen wird entfernt, indem die entsprechende Tabelle aus der Datenbank entfernt wird.

5.2.3 Verwaltung der Behandlungszentren

Im aktuellen System werden Behandlungszentren als Entitäten mit Attributen in einer Tabelle zusammengefasst. Jedes Zentrum hat eine automatisch generierte `center_id` als Primärschlüssel, den `center_code` zur Aufzählung im System, einen Namen (`center_name`) und optional Informationen über das Zentrum (`center_info`). Dies kann so beibehalten werden, da die Betrachtung der Zentren als Entitäten mit diesen vier Attributen offensichtlich sinnvoll ist.

5.2.4 Rollenverwaltung

Im aktuellen System hat jede Rolle als Entität in der Tabelle `roles` eine `role_id` als Primärschlüssel, einen Namen (`role_name`) und bestimmte Rechte (`rolerights`). Da zwei verschiedene Rollen in einem System normalerweise nicht den gleichen Namen tragen, könnte man die `role_id` streichen und den `role_name` als Primärschlüssel verwenden. Das würde etwas Speicherplatz sparen, aber in diesem Punkt die Tabelle von den anderen Tabellen des Systems unterscheiden. Beide Möglichkeiten haben also ihr Für und Wider; da Speicherplatz aber heutzutage keine rare Ressource mehr ist, ist hier die Konsistenz der Datenbank vorzuziehen. Die Tabelle kann also sinnvollerweise beibehalten werden.

Die Berechtigungen der verschiedenen Rollen werden in der Tabelle `possible_role_rights` beschrieben. Jede Berechtigung hat eine `possible_role_rights_id` als Primärschlüssel, einen Namen (`right`), und eine – hier verpflichtende – Beschreibung der Berechtigung (`right_description`). Auch

hier gilt der Vorzug der Konsistenz vor geringem Speicherplatzgewinn. Deswegen sollte die ID beibehalten werden und eine Umbenennung von `right` in `right_name` erfolgen. Ansonsten kann auch diese Tabelle in ihrer Struktur beibehalten werden.

5.2.5 Metadaten

Administratorinnen und Administratoren können bestimmte Metadaten der Datenbank verwalten. Die entsprechenden Tabellen werden nun kurz diskutiert.

In der Tabelle `score_rules` werden Regeln zur Auswertung bestimmter Fragebögen gespeichert. Wie immer wird der Primärschlüssel als automatische ID (`score_rule_id`) gespeichert. Jede Regel hat einen Namen (`score_name`) und ein Textfeld für die eigentliche Regel (`score_rule`). Weitere Informationen sind nicht erforderlich, und so kann diese Tabelle beibehalten werden.

In der Tabelle `possible_types` werden die möglichen Typen für die Tabelle `base_types` gespeichert (zum Beispiel `Fragebogen`). Jeder Typ hat eine automatisch generierte ID als Primärschlüssel (`possible_type_id`) und einen Namen (`type_name`). Auch hier sind keine Änderungen erforderlich.

`possible_sessions` zählt die möglichen Sitzungen als Vorschlagsliste für das System auf. Jede mögliche Sitzung hat eine automatisch generierte ID als Primärschlüssel (`possible_session_id`), einen Typen (`session_name`) und einen Namen (`session_name`). Wie schon bei den ersten beiden Tabellen ist auch diese intuitiv klar verständlich und kann unverändert bleiben.

`possible_a` speichert die Vorschlagsliste für unerwünschte Nebenwirkungen, `possible_adverse_events` alle weiteren Möglichkeiten. Beide enthalten eine automatisch generierte ID als Primärschlüssel, die Position der Nebenwirkung in der Auflistung beim Ausfüllen des Fragebogens und das zugehörige Behandlungszentrum. Die Tabelle
`possible_adverse_events` für konkrete mögliche Nebenwirkungen enthält zusätzlich das Attribut `event` zu ihrer Beschreibung.

Analog zu `possible_adverse_events` sind die anderen Metadatentabellen für die möglichen medizinischen Fragebogeneinträge aufgebaut. Die Tabelle `possible_comorbidity_diseases` enthält statt `event` ein Attribut `disease`, die Tabelle `possible_concomittant_medications` enthält `medication` und `possible_non_pharmological_interventions` enthält `intervention`.

Alle diese Tabellen können sinnvollerweise beibehalten werden.

5.3 Effiziente Speichernutzung mittels FOREIGN KEY Constraints

Zunächst werden die `FOREIGN KEY Constraints` von MySQL erläutert. Anschließend werden die Möglichkeiten aufgezeigt, auf entsprechende Anfragen effizient zu reagieren.

5.3.1 FOREIGN KEY Constraints in MySQL

`FOREIGN KEY Constraints` dienen in MySQL dazu, auf Veränderungen oder Entfernungen von Fremdschlüsseln zu reagieren. Die Syntaxstruktur ist in Abbildung 4.1 zu sehen.

Das Constraint wird dem Fremdschlüssel in der Kindtabelle, die den Key der Elterntabelle referenziert, zugeordnet. Dabei ist `index_name` ein optionaler Name für den

```
 [CONSTRAINT [symbol]] FOREIGN KEY
[index_name] (index_col_name, ...)
REFERENCES tbl_name (index_col_name,...)
[ON DELETE reference_option]
[ON UPDATE reference_option]

reference_option:
RESTRICT | CASCADE | SET NULL | NO ACTION
```

Abbildung 5.1: Syntaxstruktur eines FOREIGN KEY Constraints in MySQL [17]

Fremdschlüssel, um diesen im Constraint einfacher referenzieren zu können. Wenn es bereits einen definierten Index in der Kindtabelle für den Fremdschlüssel gibt, wird der `index_name` ignoriert. Wenn nicht, wird ein eventuell vorhandenes `[symbol]` aus der ersten Zeile verwendet. Ist auch dieses nicht vorhanden, wird der `index_name` verwendet. Wenn auch dieser nicht definiert wurde, wird der Name aus der refernzierenden Spalte generiert. Die Tabellenspalten, aus denen der Fremdschlüssel besteht, werden anschließend in `(index_col_name,...)` aufgezählt. Die Tabelle und die entsprechenden Spalten, die referenziert werden, sind in der nächsten Zeile `REFERENCES tbl_name (index_col_name,...)` aufgeführt. In der TDB bestehen die Schlüssel jeweils aus nur einer ID-Spalte, also werden auch die beiden Abschnitte `(index_col_name,...)` entsprechend kurz. Hierbei ist zu beachten, dass die Spaltennamen in korrespondierender Reihenfolge aufgelistet werden müssen. Schließlich enthalten die Zeilen `[ON DELETE reference_option]` und `[ON UPDATE reference_option]` die auszuführenden Aktionen. Diese sind:

- `RESTRICT`: Verweigert die Ausführung der UPDATE- beziehungsweise DELETE-Aktion in der Elterntabelle
- `CASCADE`: Verändert beziehungsweise löscht die Einträge der Kindtabelle analog zu den referenzierten Einträgen in der Elterntabelle, die verändert beziehungsweise gelöscht wurden.
- `SET NULL`: Setzt die entsprechenden FOREIGN KEYs auf NULL. Dies ist nur möglich, wenn zugehörige NOT NULL-Flags nicht gesetzt sind; da in der TDB bei allen Schlüsselattribute NOT NULL gesetzt ist, wird diese Aktion nicht verwendet
- `SET DEFAULT`: MySQL unterstützt auch das rücksetzen auf den DEFAULT-Wert, falls dieser definiert ist; dies wird allerdings von einigen Datenbankprogrammen nicht unterstützt und wird, da die IDs der TDB keine DEFAULT-Werte haben, hier ebenfalls nicht verwendet
- `NO ACTION`: Die Einträge der Kindtabelle bleiben unverändert. Da in der Datenbank auch viel mit Auswertungen und Anfragen gearbeitet wird, ist auch diese Option wenig sinnvoll, da sie ständig Fehler bei Datenbankanfragen verursachen würde.

Nicht gesetzte `ON DELETE`- beziehungsweise `ON UPDATE`-Zeilen werden als `RESTRICT` interpretiert. So ist also keine Änderung oder Löschung von referenzierten Einträgen aus der aktuellen Datenbank möglich. [17] Die MySQL-Workbench unterstützt das einfache Setzen von `FOREIGN KEY Constraints`, wenn ein Fremdschlüssel bereits gesetzt wurde (wie in der TDB). Dazu rechtsklickt man auf die entsprechende Tabelle, wählt im Kontextmenü die Option „Edit '<TABELLENNAME>'“, dann im Editiermenü den Reiter „Foreign Keys“. Dann wählt man den entsprechenden Fremdschlüssel aus und setzt unter „Foreign Key Options“ die gewünschte Option.

Veränderungen der automatisch generierten IDs sind generell unnötig, schlechter Stil und sollten nicht durchgeführt werden. Prinzipiell ist es aber kein Problem, für alle Fremdschlüsselbeziehungen ein `ON UPDATE CASCADE`-Constraint zu definieren, um im Falle einer Änderung Konsistenzprobleme zu vermeiden. Es wird vorgeschlagen, dies sicherheitshalber umzusetzen.

5.3.2 Kaskadierende Löschung nicht mehr benötigter/erwünschter Daten in der Tinnitus Database

Wir betrachten zunächst eine Beispielsituation: Für die Löschung eines Benutzerkontos gibt es zwei Möglichkeiten:

- Die Benutzerin/der Benutzer möchte ihr/sein Konto entfernen und von ihr/ihm behandelte Patientinnen/Patienten und deren Daten sollen aus dem System entfernt werden.
- Die Benutzerin/der Benutzer möchte ihr/sein Konto entfernen, aber von ihr/ihm behandelte Patientinnen/Patienten und deren Daten sollen im System verbleiben.

Für den ersten Fall muss ein `FOREIGN KEY Constraint` mit der Zeile `ON DELETE CASCADE` für den Fremdschlüssel `creator_id` in der Tabelle `patients` definiert werden. Dann werden die entsprechenden Patientinnen/Patienten und deren Daten aus dem System gelöscht.

Im zweiten Fall muss ein user-Eintrag in der Datenbank verbleiben, da sonst das vorher definierte Constraint die Patientendaten entfernen würde. Um die Daten des

entsprechenden Nutzers dennoch zu entfernen, kann man sie auf Leerwerte setzen (0 für Zahlen, leerer String für Texte, 1. 1. 1970 für Datumsvariablen etc.). So bleiben die Patientendaten erhalten. Die ID der Benutzerin/des Benutzers, die erhalten bleiben muss, wird automatisch fortlaufend generiert und enthält daher keine Informationen über die Benutzerin/den Benutzer. Der referenzierte Eintrag wird also nicht tatsächlich gelöscht, aber ausreichend unbrauchbar gemacht. Es wird also im System eine Abfrage benötigt, ob die Daten verbleiben sollen. Wenn ja, wird der Tabelleneintrag nicht gelöscht und damit das `FOREIGN KEY Constraint` nicht ausgelöst.

Analog können Fremdschlüssel, die die Tabellen `sessions`, `patients` und `patient_records` referenzieren, behandelt werden. Da das Geschlecht und das Alter der Patientinnen/Patienten für Statistiken, Datenauswertung und -analyse relevant ist, aber eventuell Rückschlüsse auf Personen zulässt, sollte hier im Einzelfall der Wille der Patientin/des Patienten erfragt werden. Die `ON DELETE`-Zeilen der Tabellen `base_types` und `roles` sollten auf `RESTRICT` verbleiben, da undefinierte Typen beziehungsweise Rollen für Probleme bei der Systemnutzung sorgen können. Bei allen weiteren Fremdschlüsselbeziehungen kann `ON DELETE CASCADE` definiert werden, da zum Beispiel das Entfernen der Metatabelle eines Fragebogens bedeutet, dass der Fragebogen nicht mehr verwendet wird. Die entsprechenden Daten können also aus dem System entfernt werden. Zu beachten ist hierbei, dass eine solche Löschung nur durchgeführt werden sollte, wenn der entsprechende Fragebogen tatsächlich nicht mehr benötigt wird, weder zur Datensicherung und -analyse, noch für zukünftige Behandlungen.

5.4 Trigger für E-Mail-Benachrichtigungen bei Änderungen in der Datenbank

In diesem Abschnitt wird zunächst ein Überblick über die Funktion von MySQL-Triggern gegeben. Anschließend werden Trigger entwickelt, die E-Mail-Benachrichtigungen im Fall von Datensatz-Veränderungen unterstützen. Dabei wird erläutert, dass eine direkte Versendung der E-Mails über den Datenbankserver nicht förderlich ist. Stattdessen kann

```
 CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }
```

Abbildung 5.2: Syntaxstruktur eines CREATE TRIGGER Statements in MySQL [18]

zum Beispiel ein separater Skriptserver von den Triggern befüllte Tabellen in festlegbaren Abständen abfragen und entsprechend den Einstellungen Mails verschicken.

5.4.1 MySQL-Trigger

Ein Trigger ist ein benamtes, mit einer Tabelle assoziiertes Datenbankobjekt, das aktiviert wird, wenn in der Tabelle ein bestimmtes Ereignis auftritt. Die Syntaxstruktur eines `CREATE TRIGGER` Statements ist in Abbildung 4.2 zu sehen.

Trigger können erstellt und entfernt werden. Beim Entfernen wird das Statement mit `DROP` begonnen, beim Erstellen mit `CREATE`. Es kann optional ein `DEFINER`-Wert angegeben werden; diese Benutzerin/dieser Benutzer muss die `TRIGGER`-Berechtigung oder eine höhere haben, welche diese einschließt, damit der Trigger erstellt werden kann. Der Default-Wert ist `CURRENT_USER`. Der `trigger_name` muss explizit angegeben werden. In der nächsten Zeile wird bestimmt, wann der Trigger feuert. Dies kann vor oder nach einer Aktion der Fall sein (`trigger_time = BEFORE` beziehungsweise `trigger_time = AFTER`), wobei eine Aktion entweder ein Hinzufügen (`INSERT`) neuer Reihen oder eine Veränderung (`UPDATE`) oder Löschung (`DELETE`) bestehender Reihen ist. Zu beachten ist, dass ein `ON DELETE`-Trigger nicht feuert, wenn die ganze Tabelle gelöscht (`DROP TABLE`) oder geleert (`TRUNCATE TABLE`) wird. `ON tbl_name FOR EACH ROW` referenziert die Tabelle, der der Trigger zugeordnet ist. Jeder Tabellen-

eintrag, der das `trigger_event` ausgelöst hat, wird bearbeitet. Im `trigger_body` wird schließlich beschrieben, was der Trigger tun soll. Mit `BEGIN` und `END` können hier mehrere Befehle ausgeführt werden, werden sie weggelassen, so kann nur ein Befehl ausgeführt werden. Es kann mit `OLD` und `NEW` auf den Zustand der Tabelle vor und nach dem Ereignis referenziert werden. Die Befehle sind Datenbankanfragen, die sich auch mittels `SET` direkt auf einzelne Werte beziehen können. Auch andere Tabellen können in Triggern referenziert werden, vorausgesetzt die nötigen Berechtigungen sind vorhanden.

5.4.2 Entwurf eines Modells für E-Mail-Benachrichtigungen

Damit E-Mail-Benachrichtigungen erfolgen können, müssen zunächst Veränderungen der Datensätze bekannt gemacht werden. Diese werden in einer eigenen Tabelle `data_changes` gespeichert. Wie muss diese aussehen? Damit Benachrichtigungen sinnvoll gefiltert werden können, muss klar sein, welche Aktion auf welchem Eintrag durchgeführt wurde. Die Tabelle `data_changes` muss also die durchgeführte Aktion, den Namen der Tabelle, welche verändert wurde, die ID des geänderten Eintrags, die alten und neuen Eintragswerte in komprimierter Form und den Änderungszeitpunkt enthalten. Hierfür werden also sechs Spalten benötigt:

- `action ENUM ('added','changed','removed')`
- `table_name TEXT`
- `entry_id INT`
- `old_value TEXT`
- `new_value TEXT`
- `change_time TIMESTAMP`

Da Daten auch hinzugefügt oder gelöscht werden können, wird das Flag `NOT NULL` bei `old_value` und `new_value` nicht gesetzt, damit sie leerbleiben können; bei den anderen Attributen wird es gesetzt.

Um nun diese Tabelle zu befüllen, benötigt man die eben erwähnten Trigger. Beispielhaft ist in Abbildung 4.3 ein Trigger für das Einfügen eines neuen Patienten angeführt.

```
 CREATE
TRIGGER new_patient
AFTER INSERT
ON patients FOR EACH ROW
INSERT INTO data_changes ('added', 'patients', NEW.patient_id,
NULL, GROUP_CONCAT(NEW.patient_group, NEW.creator_id,
NEW.dateofbirth, NEW.sex), CURRENT_TIMESTAMP)
```

Abbildung 5.3: Beispiel für einen Trigger zur Abbildung von Datensatzänderungen

Analog kann jede Tabelle der Datenbank mit Triggern versehen werden, damit Benutzerinnen/Benutzer sich über Veränderungen benachrichtigen lassen können.

Angenommen, eine Benutzerin/ein Benutzer möchte nun per E-Mail informiert werden, wenn bestimmte Änderungen in der Datenbank vorgenommen werden. Dazu sind die E-Mail-Adresse der Benutzerin/des Benutzers, die Abstände, in denen die Information erfolgen soll (zum Beispiel jeden Tag, einmal die Woche etc.), und die Datensätze, bei deren Änderung eine Benachrichtigung erfolgen soll, erforderlich. Diese werden in einer Tabelle `notifications` gespeichert, die entsprechende Attribute enthält:

- `email VARCHAR(255)`
- `interval ENUM('instantly','daily','weekly','monthly')`
- `data TEXT`

Alle Attribute sind zwingend erforderlich, das `NOT NULL` Flag wird also immer gesetzt. Die E-Mail-Adresse kann beim Einstellen einer E-Mail-Benachrichtigung aus der Tabelle `users` abgefragt werden. Je nach Umsetzung können die Möglichkeiten für die Benachrichtigungsabstände durch Änderung der Werte von `interval` leicht geändert werden. Das Attribut `data` muss nun die Änderungen enthalten, nach denen die Tabelle `data_changes` gefiltert werden soll. Eine Möglichkeit zur Strukturierung dieses Eintrags wäre die Abtrennung verschiedener Benachrichtigungen mit Semikolons und die Abtrennung der Filter mit Kommata. Ein Beispiel: `added, patients, dateofbirth > 1980-05-05, sex = m; removed, patients, sex = f.`

Hiermit existiert die Möglichkeit, in der Datenbank Änderungen zu speichern und Filter für E-Mail-Benachrichtigungen zu hinterlegen. Nun müssen diese noch ausgewertet und die E-Mails verschickt werden.

5.4.3 Problem beim Verschicken von E-Mails direkt auf dem Datenbankserver, Alternative: Skriptserver

Das Problem, wenn der Datenbankserver selbst die E-Mails verschickt, ist eine mögliche Überlastung. Wenn viele Updates zeitnah passieren, die E-Mails an viele Benutzerinnen/Benutzer erfordern, kann dies soweit gehen, dass der Datenbankserver Anfragen nicht mehr schnell genug verarbeiten kann und die Echtzeitanforderung aus 4.3 nicht mehr erfüllt wird. Um dies zu verhindern, ist es sinnvoll, die Verarbeitung der E-Mail-Tabellendaten einem Skriptserver zu überlassen. Dieser kann in regelmäßigen Abständen die Einträge abgreifen und entsprechend E-Mails verschicken. Dies ermöglicht zudem, dass Benutzerinnen/Benutzer verschiedene Benachrichtigungsabstände einstellen können, die der Skriptserver dann beachten kann. Auch eine Sofort-Benachrichtigungs-Option wäre möglich, wenn man einen Listener implementiert, der bei Veränderungen der Tabelle `data_changes` reagiert.

6

Abgleich der Anforderungen

In diesem Abschnitt wird überprüft, ob die Anforderungen aus Kapitel 4 in der Analyse in Kapitel 5 erfüllt wurden.

6.1 Grundlegende Systemfunktionen

Die Registrierung und Entfernung einer Benutzerin/eines Benutzers erfolgt über entsprechende Änderungen in der Tabelle `users`. Patientinnen und Patienten werden in den Tabellen `patients` und `id_translation` verwaltet. Die Löschung von Daten aus `id_translation` erfolgt gegebenenfalls über `FOREIGN KEY Constraints`. Analog erfolgt die Verwaltung von Behandlungssitzungen, gegebenenfalls mit kaskadierender Löschung von Tabelleneinträgen, die die Tabelle `sessions` mit `FOREIGN KEY Constraints` referenzieren. Medizinische Untersuchungsdaten und die Daten

aus den Fragebögen können in die entsprechenden Tabellen eingetragen werden.(siehe Abschnitt 5.1)

Statistiken sind mittels Anfragen an die Datenbank abrufbar; die Darstellung wird vom System übernommen und hat keinen Einfluss auf die Struktur der Datenbank.

E-Mail-Benachrichtigungen werden mittels der Tabellen `email_notifications` und `data_changes` realisiert.(siehe Abschnitt 5.4)

6.2 Systemadministration

Administratorinnen und Administratoren können Fragebögen, Behandlungszentren, Benutzer und deren Rollen sowie Metadaten in der Datenbank mittels der dafür vorgesehenen Tabellen verwalten (siehe Abschnitt 5.2).

In Abschnitt 5.2.1 wurde eine Möglichkeit beschrieben, eine Validierung von Fragebögen in der Datenbank umzusetzen.

6.3 Nicht-funktionale Anforderungen

Mit einem Update auf Version 5.6 kann MySQL beibehalten werden. Dies kann theoretisch auch auf dem Webserver der TDB problemlos umgesetzt werden.

Ein aktualisiertes ER-Diagramm zu Dokumentationszwecken ist in Anhang B verfügbar.

Durch die Auslagerung der E-Mail-Benachrichtigungen an den Skriptserver wird der Datenbankserver mit den E-Mail-Benachrichtigungen kaum mehr belastet und sollte in der Lage sein, (fast) genauso schnell auf Anfragen zu reagieren wie ohne E-Mail-Benachrichtigungen.(siehe Abschnitt 5.4.3)

7

Zusammenfassung und Fazit

In dieser Arbeit wurde eine Funktionsanalyse der Tinnitus Database des TRI durchgeführt.

Zunächst wurde ein Anforderungskatalog erstellt, um bestehende und erwünschte Funktionalitäten und Eigenschaften der Datenbank beziehungsweise des Systems zu erfassen.

In der Analyse wurden mögliche Änderungen erläutert. Die Einbindung eines Validierungssystems wurde vorgeschlagen. Ebenso wurde erläutert, wie die Funktionalität der Datenbank durch FOREIGN KEY Constraints zur effizienteren Datenhaltung erweitert werden kann. Schließlich wurden Tabellen zur datenbankinternen Abbildung von Daten für die Benachrichtigung von Benutzerinnen und Benutzern per E-Mail über Veränderungen in der Datenbank entworfen.

Es hat sich gezeigt, dass alle Anforderungen theoretisch erfüllt sind. Die praktische Umsetzung der Entwürfe dieser Arbeit und die Auslotung eventueller weiterer Möglichkeiten, die besprochenen MySQL-Funktionen zu nutzen, bleibt nachfolgenden Projekten überlassen.

Ein weiteres Projekt, das bereits umgesetzt wurde, ist „Track Your Tinnitus“, eine App, die die Belastung durch Tinnitus im Alltag mittels Fragebögen aufzeichnet. [19] [20] [21]

Literaturverzeichnis

[1] Hagenlocher, R.: Designkonzept für eine Webanwendung zum Zugriff auf eine interdisziplinäre und multinationale Datenbank zur Erfassung Tinnitus-geschädigter Patienten. Masterarbeit, Universität Ulm (2015)

[2] o.V.: About us (2016) Informationen über die Datenbank des TRI, zuletzt abgerufen am 09.08.2016.

[3] o.V.: About COST (2016) Informationen über die COST-Initiative, zuletzt abgerufen am 09.08.2016.

[4] o.V.: BMBS COST Action BM1306 (2016) Better Understanding the Heterogeneity of Tinnitus to Improve and Develop New Treatments (TINNET), zuletzt abgerufen am 09.08.2016.

[5] Zeman, F., Koller, M., Schecklmann, M., Langguth, B., Landgrebe, M.: Tinnitus assessment by means of standardized self-report questionnaires: Psychometric properties of the Tinnitus Questionnaire (TQ), the Tinnitus Handicap Inventory (THI), and their short versions in an international and multi-lingual sample. Health and Quality of Life Outcomes **10** (2012) 1–10

[6] Schecklmann, M., Landgrebe, M., Langguth, B., the TRI Database Study Group: Phenotypic Characteristics of Hyperacusis in Tinnitus. PLoS ONE **9** (2014) 1–7

[7] Stenske, I.: Entwicklung eines Design-Konzepts für eine multinationale Forschungsdatenbank zur Speicherung von longitudinalen Patientendaten. (2015)

[8] Baltakiranoglu, B.: Konzeption und Realisierung eines Patientenmoduls für eine multinationale und interdisziplinäre Datenbank. (2015)

[9] Landgrebe, M., Zeman, F., Koller, M., Eberl, Y., Mohr, M., Reiter, J., Staudinger, S., Hajak, G., Langguth, B.: The Tinnitus Research Initiative (TRI) database: A new approach for delineation of tinnitus subtypes and generation of predictors for treatment outcome. BMC Medical Informatics and Decision Making **10** (2010) 1–7

[10] Unland, R., Pernul, G.: Datenbanken im Einsatz: Analyse, Modellbildung und Umsetzung. Walter de Gruyter GmbH & Co KG (2014)

[11] Gnad, F., Parsch, J.: Sebida: a database for the functional and evolutionary analysis of genes with sex-biased expression. Bioinformatics **22** (2006) 2577–2579

[12] Mitchell, T.D., Jones, P.D.: An improved method of constructing a database of monthly climate observations and associated high-resolution grids. International Journal of Climatology **25** (2005) 693–712

[13] Grzybicki, D.M., Turcsany, B., Becich, M.J., Gupta, D., Gilbertson, J.R., Raab, S.S.: Database Construction for Improving Patient Safety by Examining Pathology Errors. American Journal of Clinical Pathology **124** (2005) 500–509

[14] o.V.: Automatic Initialization and Updating for TIMESTAMP and DATETIME (2016) MySQL 5.6 Reference Manual, zuletzt abgerufen am 10.08.2016.

[15] o.V.: Automatic Initialization and Updating for TIMESTAMP and DATETIME (2016) MySQL 5.5 Reference Manual, zuletzt abgerufen am 10.08.2016.

[16] o.V.: Numeric Type Overview (2016) MySQL 5.6 Reference Manual, zuletzt abgerufen am 12.08.2016.

[17] o.V.: Using FOREIGN KEY Constraints (2016) MySQL 5.6 Reference Manual, zuletzt abgerufen am 12.08.2016.

[18] o.V.: CREATE TRIGGER Syntax (2016) MySQL 5.6 Reference Manual, zuletzt abgerufen am 13.08.2016.

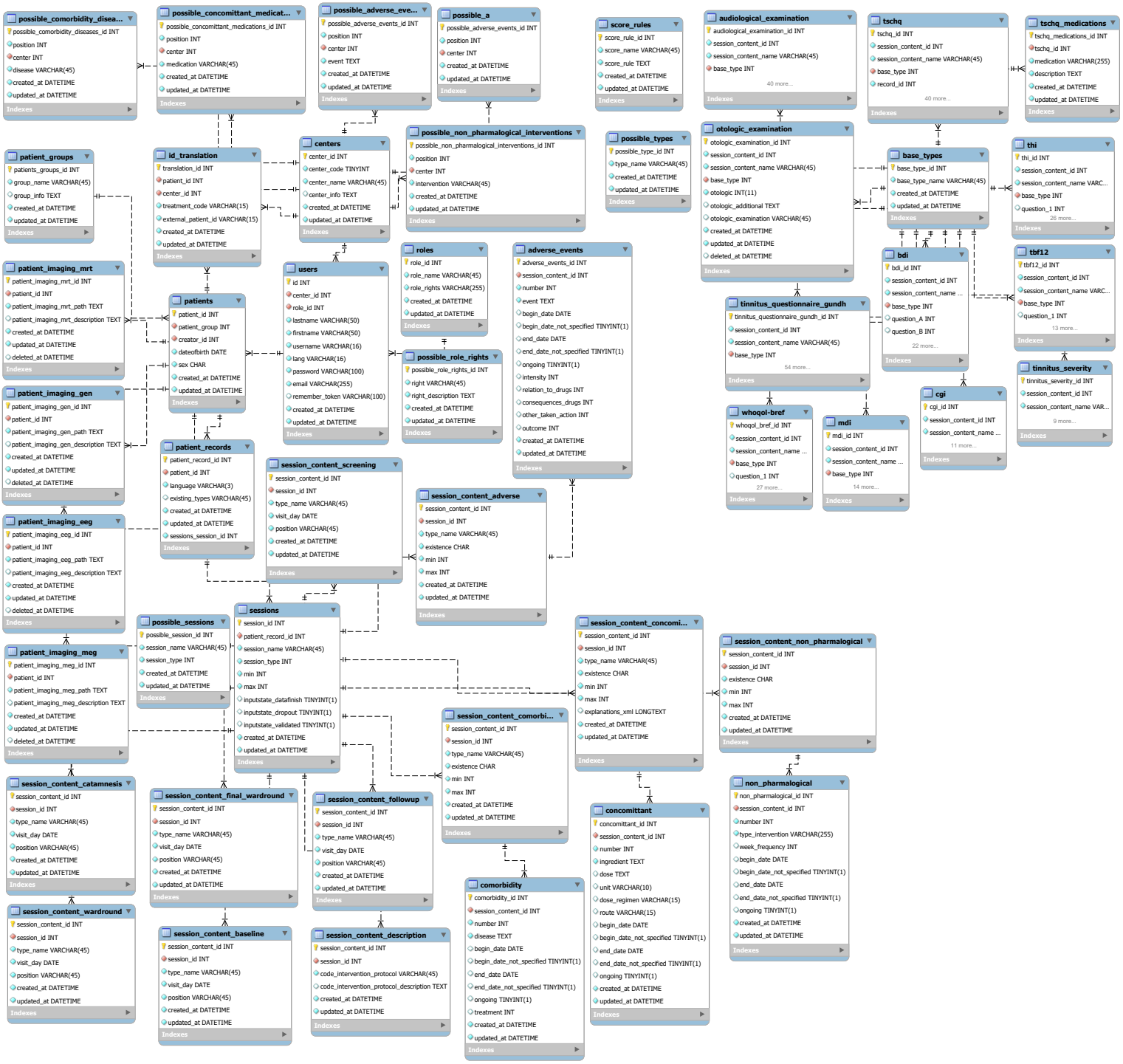
[19] Pryss, R., Reichert, M., Langguth, B., Schlee, W.: Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy and Research. In: IEEE 4th International Conference on Mobile Services (MS 2015), IEEE Computer Society Press (2015) 352–359

[20] Pryss, R., Reichert, M., Herrmann, J., Langguth, B., Schlee, W.: Mobile Crowd Sensing in Clinical and Psychological Trials ? A Case Study. In: 28th IEEE Int'l Symposium on Computer-Based Medical Systems, IEEE Computer Society Press (2015) 23–24

[21] Probst, T., Pryss, R., Langguth, B., Schlee, W.: Emotional states as mediators between tinnitus loudness and tinnitus distress in daily life: Results from the "TrackYourTinnitus" application. Scientific Reports **6** (2016)

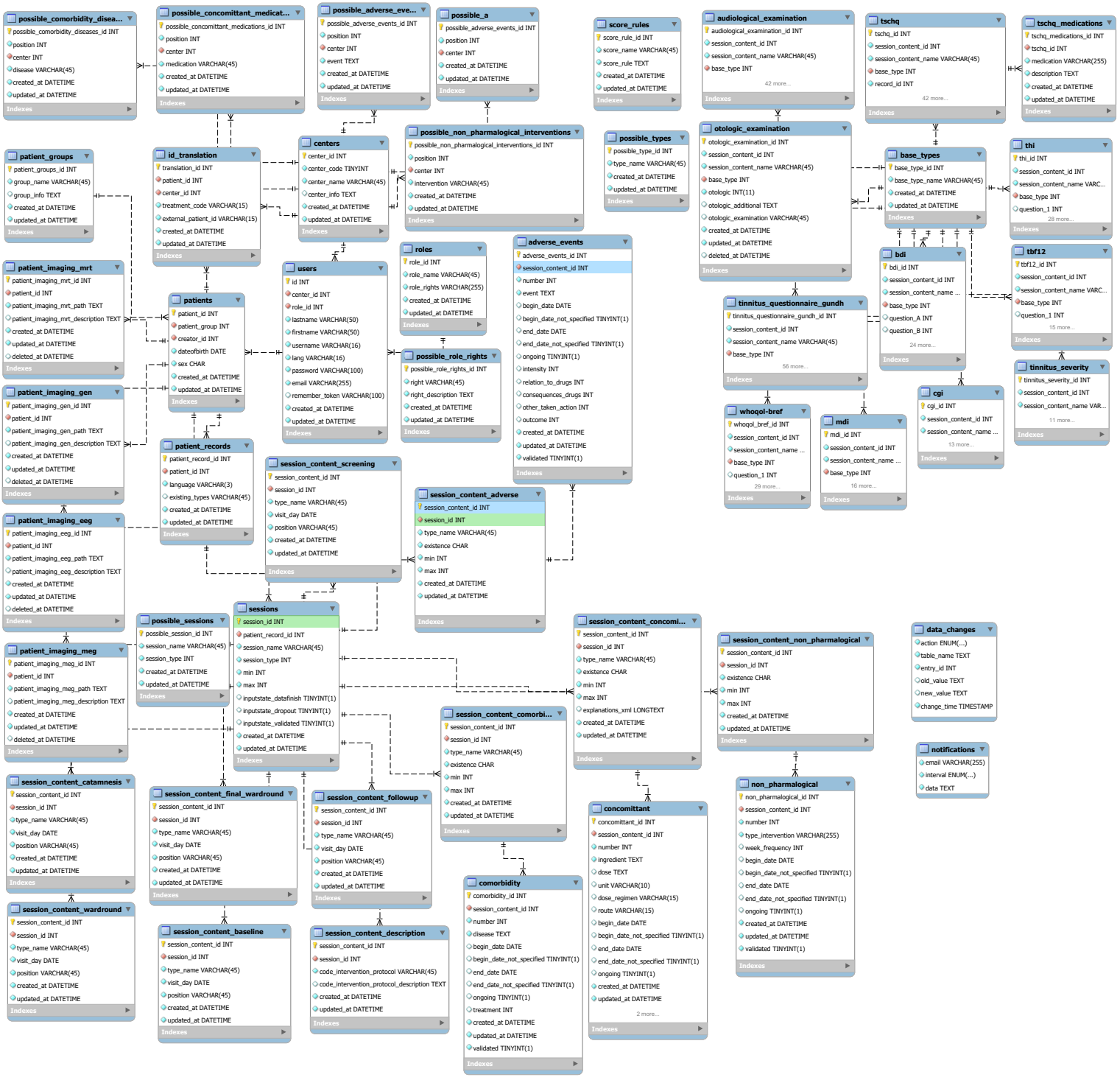
A

Aktuelles ER-Diagramm der Tinnitus Database


possible_comorbidity_disea...
possible_comorbidity_diseases_id INT
position INT
center INT
disease VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_concomittant_medicat...
possible_concomittant_medications_id INT
position INT
center INT
medication VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_adverse_eve...
possible_adverse_events_id INT
position INT
center INT
event TEXT
created_at DATETIME
updated_at DATETIME
Indexes
possible_a
possible_adverse_events_id INT
position INT
center INT
created_at DATETIME
updated_at DATETIME
Indexes
score_rules
score_rule_id INT
score_name VARCHAR(45)
score_rule TEXT
created_at DATETIME
updated_at DATETIME
Indexes
audiological_examination
audiological_examination_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
40 more...
Indexes
tschq
tschq_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
record_id INT
40 more...
Indexes
tschq_medications
tschq_medications_id INT
tschq_id INT
medication VARCHAR(255)
description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
patient_groups
patients_groups_id INT
group_name VARCHAR(45)
group_info TEXT
created_at DATETIME
updated_at DATETIME
Indexes
id_translation
translation_id INT
patient_id INT
center_id INT
treatment_code VARCHAR(15)
external_patient_id VARCHAR(15)
created_at DATETIME
updated_at DATETIME
Indexes
centers
center_id INT
center_code TINYINT
center_name VARCHAR(45)
center_info TEXT
created_at DATETIME
updated_at DATETIME
Indexes
possible_non_pharmalogical_interventions
possible_non_pharmalogical_interventions_id INT
position INT
center INT
intervention VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_types
possible_type_id INT
type_name VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
otologic_examination
otologic_examination_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
otologic INT(11)
otologic_additional TEXT
otologic_examination VARCHAR(45)
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
base_types
base_type_id INT
base_type_name VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
thi
thi_id INT
session_content_id INT
session_content_name VARC...
base_type INT
question_1 INT
26 more...
Indexes
patient_imaging_mrt
patient_imaging_mrt_id INT
patient_id INT
patient_imaging_mrt_path TEXT
patient_imaging_mrt_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
patients
patient_id INT
patient_group INT
creator_id INT
dateofbirth DATE
sex CHAR
created_at DATETIME
updated_at DATETIME
Indexes
users
id INT
center_id INT
role_id INT
lastname VARCHAR(50)
firstname VARCHAR(50)
username VARCHAR(16)
lang VARCHAR(16)
password VARCHAR(100)
email VARCHAR(255)
remember_token VARCHAR(100)
created_at DATETIME
updated_at DATETIME
Indexes
roles
role_id INT
role_name VARCHAR(45)
role_rights VARCHAR(255)
created_at DATETIME
updated_at DATETIME
Indexes
possible_role_rights
possible_role_rights_id INT
right VARCHAR(45)
right_description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
adverse_events
adverse_events_id INT
session_content_id INT
number INT
event TEXT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
intensity INT
relation_to_drugs INT
consequences_drugs INT
other_taken_action INT
outcome INT
created_at DATETIME
updated_at DATETIME
Indexes
tinnitus_questionnaire_gundh
tinnitus_questionnaire_gundh_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
54 more...
Indexes
bdi
bdi_id INT
session_content_id INT
session_content_name ...
base_type INT
question_A INT
question_B INT
22 more...
Indexes
tbf12
tbf12_id INT
session_content_id INT
session_content_name VARC...
base_type INT
question_1 INT
13 more...
Indexes
tinnitus_severity
tinnitus_severity_id INT
session_content_id INT
session_content_name VAR...
9 more...
Indexes
cgi
cgi_id INT
session_content_id INT
session_content_name ...
11 more...
Indexes
whoqol-bref
whoqol_bref_id INT
session_content_id INT
session_content_name ...
base_type INT
question_1 INT
27 more...
Indexes
mdi
mdi_id INT
session_content_id INT
session_content_name ...
base_type INT
14 more...
Indexes
patient_imaging_gen
patient_imaging_gen_id INT
patient_id INT
patient_imaging_gen_path TEXT
patient_imaging_gen_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
patient_records
patient_record_id INT
patient_id INT
language VARCHAR(3)
existing_types VARCHAR(45)
created_at DATETIME
updated_at DATETIME
sessions_session_id INT
Indexes
session_content_screening
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_adverse
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
patient_imaging_eeg
patient_imaging_eeg_id INT
patient_id INT
patient_imaging_eeg_path TEXT
patient_imaging_eeg_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
possible_sessions
possible_session_id INT
session_name VARCHAR(45)
session_type INT
created_at DATETIME
updated_at DATETIME
Indexes
sessions
session_id INT
patient_record_id INT
session_name VARCHAR(45)
session_type INT
min INT
max INT
inputstate_datafinish TINYINT(1)
inputstate_dropout TINYINT(1)
inputstate_validated TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_concomi...
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
explanations_xml LONGTEXT
created_at DATETIME
updated_at DATETIME
Indexes
session_content_non_pharmalogical
session_content_id INT
session_id INT
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
patient_imaging_meg
patient_imaging_meg_id INT
patient_id INT
patient_imaging_meg_path TEXT
patient_imaging_meg_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
session_content_comorbi...
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
session_content_catamnesis
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_final_wardround
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_followup
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
concomittant
concomittant_id INT
session_content_id INT
number INT
ingredient TEXT
dose TEXT
unit VARCHAR(10)
dose_regimen VARCHAR(15)
route VARCHAR(15)
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
non_pharmalogical
non_pharmalogical_id INT
session_content_id INT
number INT
type_intervention VARCHAR(255)
week_frequency INT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_wardround
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_baseline
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_description
session_content_id INT
session_id INT
code_intervention_protocol VARCHAR(45)
code_intervention_protocol_description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
comorbidity
comorbidity_id INT
session_content_id INT
number INT
disease TEXT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
treatment INT
created_at DATETIME
updated_at DATETIME
Indexes

B

Neues ER-Diagramm


possible_comorbidity_disea...
possible_comorbidity_diseases_id INT
position INT
center INT
disease VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_concomittant_medicat...
possible_concomittant_medications_id INT
position INT
center INT
medication VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_adverse_eve...
possible_adverse_events_id INT
position INT
center INT
event TEXT
created_at DATETIME
updated_at DATETIME
Indexes
possible_a
possible_adverse_events_id INT
position INT
center INT
created_at DATETIME
updated_at DATETIME
Indexes
score_rules
score_rule_id INT
score_name VARCHAR(45)
score_rule TEXT
created_at DATETIME
updated_at DATETIME
Indexes
audiological_examination
audiological_examination_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
42 more...
Indexes
tschq
tschq_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
record_id INT
42 more...
Indexes
tschq_medications
tschq_medications_id INT
tschq_id INT
medication VARCHAR(255)
description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
patient_groups
patient_groups_id INT
group_name VARCHAR(45)
group_info TEXT
created_at DATETIME
updated_at DATETIME
Indexes
id_translation
translation_id INT
patient_id INT
center_id INT
treatment_code VARCHAR(15)
external_patient_id VARCHAR(15)
created_at DATETIME
updated_at DATETIME
Indexes
centers
center_id INT
center_code TINYINT
center_name VARCHAR(45)
center_info TEXT
created_at DATETIME
updated_at DATETIME
Indexes
possible_non_pharmalogical_interventions
possible_non_pharmalogical_interventions_id INT
position INT
center INT
intervention VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
possible_types
possible_type_id INT
type_name VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
otologic_examination
otologic_examination_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
otologic INT(11)
otologic_additional TEXT
otologic_examination VARCHAR(45)
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
base_types
base_type_id INT
base_type_name VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
thi
thi_id INT
session_content_id INT
session_content_name VARC...
base_type INT
question_1 INT
28 more...
Indexes
patient_imaging_mrt
patient_imaging_mrt_id INT
patient_id INT
patient_imaging_mrt_path TEXT
patient_imaging_mrt_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
patients
patient_id INT
patient_group INT
creator_id INT
dateofbirth DATE
sex CHAR
created_at DATETIME
updated_at DATETIME
Indexes
users
id INT
center_id INT
role_id INT
lastname VARCHAR(50)
firstname VARCHAR(50)
username VARCHAR(16)
lang VARCHAR(16)
password VARCHAR(100)
email VARCHAR(255)
remember_token VARCHAR(100)
created_at DATETIME
updated_at DATETIME
Indexes
roles
role_id INT
role_name VARCHAR(45)
role_rights VARCHAR(255)
created_at DATETIME
updated_at DATETIME
Indexes
possible_role_rights
possible_role_rights_id INT
right VARCHAR(45)
right_description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
adverse_events
adverse_events_id INT
session_content_id INT
number INT
event TEXT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
intensity INT
relation_to_drugs INT
consequences_drugs INT
other_taken_action INT
outcome INT
created_at DATETIME
updated_at DATETIME
validated TINYINT(1)
Indexes
tinnitus_questionnaire_gundh
tinnitus_questionnaire_gundh_id INT
session_content_id INT
session_content_name VARCHAR(45)
base_type INT
56 more...
Indexes
bdi
bdi_id INT
session_content_id INT
session_content_name ...
base_type INT
question_A INT
question_B INT
24 more...
Indexes
tbf12
tbf12_id INT
session_content_id INT
session_content_name VARC...
base_type INT
question_1 INT
15 more...
Indexes
tinnitus_severity
tinnitus_severity_id INT
session_content_id INT
session_content_name VAR...
11 more...
Indexes
patient_imaging_gen
patient_imaging_gen_id INT
patient_id INT
patient_imaging_gen_path TEXT
patient_imaging_gen_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
patient_records
patient_record_id INT
patient_id INT
language VARCHAR(3)
existing_types VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_screening
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_adverse
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
whoqol-bref
whoqol_bref_id INT
session_content_id INT
session_content_name ...
base_type INT
question_1 INT
29 more...
Indexes
mdi
mdi_id INT
session_content_id INT
session_content_name ...
base_type INT
16 more...
Indexes
cgi
cgi_id INT
session_content_id INT
session_content_name ...
13 more...
Indexes
patient_imaging_eeg
patient_imaging_eeg_id INT
patient_id INT
patient_imaging_eeg_path TEXT
patient_imaging_eeg_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
possible_sessions
possible_session_id INT
session_name VARCHAR(45)
session_type INT
created_at DATETIME
updated_at DATETIME
Indexes
sessions
session_id INT
patient_record_id INT
session_name VARCHAR(45)
session_type INT
min INT
max INT
inputstate_datafinish TINYINT(1)
inputstate_dropout TINYINT(1)
inputstate_validated TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_concomi...
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
explanations_xml LONGTEXT
created_at DATETIME
updated_at DATETIME
Indexes
session_content_non_pharmalogical
session_content_id INT
session_id INT
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
data_changes
action ENUM(...)
table_name TEXT
entry_id INT
old_value TEXT
new_value TEXT
change_time TIMESTAMP
patient_imaging_meg
patient_imaging_meg_id INT
patient_id INT
patient_imaging_meg_path TEXT
patient_imaging_meg_description TEXT
created_at DATETIME
updated_at DATETIME
deleted_at DATETIME
Indexes
session_content_comorbi...
session_content_id INT
session_id INT
type_name VARCHAR(45)
existence CHAR
min INT
max INT
created_at DATETIME
updated_at DATETIME
Indexes
notifications
email VARCHAR(255)
interval ENUM(...)
data TEXT
session_content_catamnesis
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_final_wardround
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_followup
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
concomittant
concomittant_id INT
session_content_id INT
number INT
ingredient TEXT
dose TEXT
unit VARCHAR(10)
dose_regimen VARCHAR(15)
route VARCHAR(15)
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
created_at DATETIME
updated_at DATETIME
2 more...
Indexes
non_pharmalogical
non_pharmalogical_id INT
session_content_id INT
number INT
type_intervention VARCHAR(255)
week_frequency INT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
created_at DATETIME
updated_at DATETIME
validated TINYINT(1)
Indexes
session_content_wardround
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_baseline
session_content_id INT
session_id INT
type_name VARCHAR(45)
visit_day DATE
position VARCHAR(45)
created_at DATETIME
updated_at DATETIME
Indexes
session_content_description
session_content_id INT
session_id INT
code_intervention_protocol VARCHAR(45)
code_intervention_protocol_description TEXT
created_at DATETIME
updated_at DATETIME
Indexes
comorbidity
comorbidity_id INT
session_content_id INT
number INT
disease TEXT
begin_date DATE
begin_date_not_specified TINYINT(1)
end_date DATE
end_date_not_specified TINYINT(1)
ongoing TINYINT(1)
treatment INT
created_at DATETIME
updated_at DATETIME
validated TINYINT(1)
Indexes

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Dominik Könke Matrikelnummer: 787241

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Dominik Könke