Technical Description of JCRdb (Jena Clinical Research Database) v1.0

Stefan Brodoehl (sb)

Oct 18, 2023

CONTENT:

CHAPTER

ONE

1. MOTIVATION:

To provide a user-friendly, clean interface for the storage, retrieval, and visualization of multimodal clinical data at the Jena University Hospital.

CHAPTER

TWO

2. OVERVIEW:

The research database is specifically engineered for seamless integration into the desktop workspaces at Jena University Hospital. It features a streamlined, intuitive frontend for user interaction, facilitating the storage, retrieval, and visualization of multimodal clinical data. The backend employs a relational database management system optimized for efficient data handling.

Due to stringent local IT infrastructure constraints, driven primarily by the need to adhere to General Data Protection Regulations (GDPR), the database resides within the secure local network of Jena University Hospital. It is encapsulated in a single SQLite file, ensuring ease of management and high-speed access. User authentication is mandatory for database access, and connections are limited to those originating from within the hospital's local network.

The frontend operates independently from a self-contained folder and requires no installation. To interact with the SQLite database, users must have read and write permissions. Importantly, all data remains securely stored in the backend database, while the frontend serves solely as an access and manipulation tool.

The database is available to a group of staff at the University Hospital of Jena. It is not available to the public. Multiple instances (i.e. database files) can be used to manage different projects independently. The database is designed to be easily extensible, allowing the addition of new data types and functionalities. Data can also be exchanged with other database instances or other systems using a defined HL7 JSON file standard.

The following figure illustrates the principle architecture of the database:

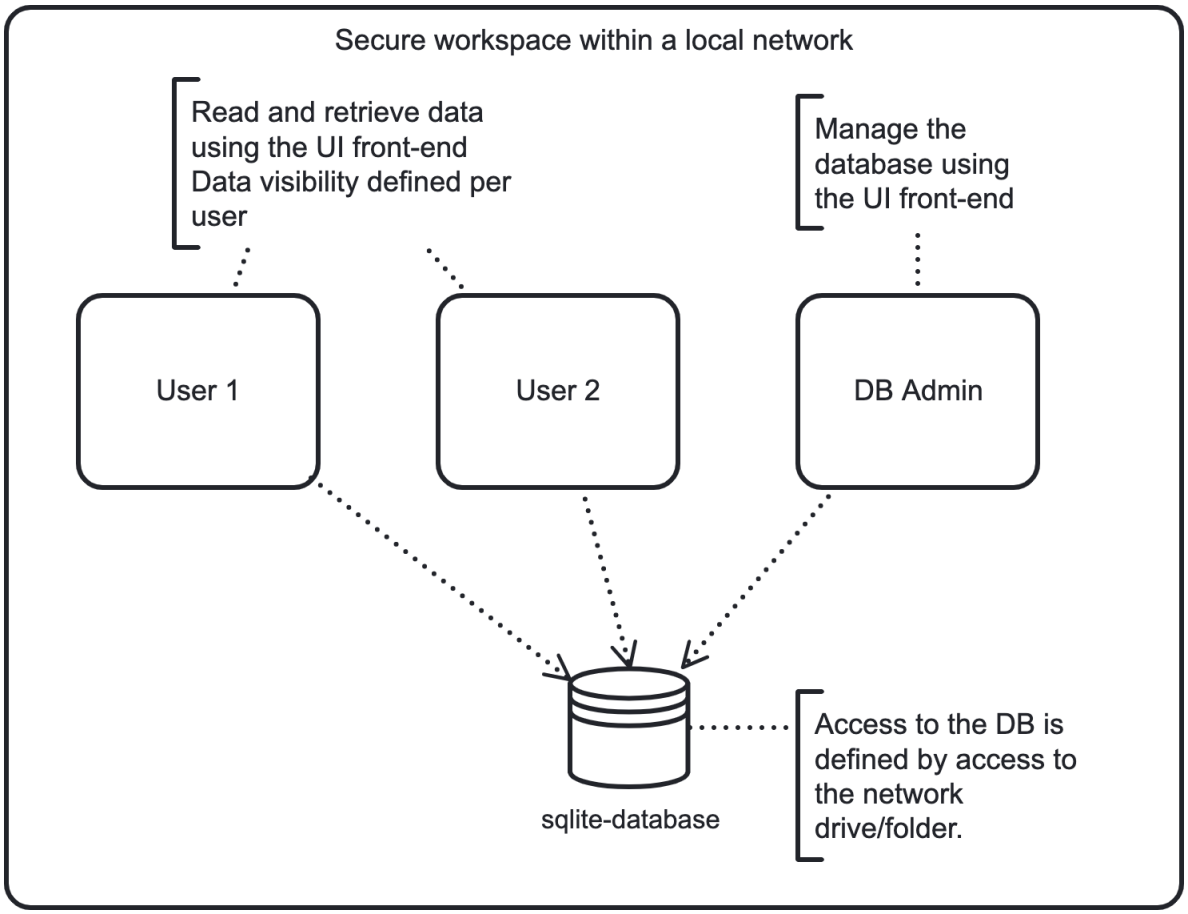


Fig. 1: Overview to the database employed at the Jena University Hospital

CHAPTER

THREE

3. ARCHITECTURE:

Database templates and build software are available at: [GitHub](https://github.com/stebro01/research_database_sqlite_i2b2.git)

3.1 Frontend

- Technology: Developed using the Quasar/Vue.js 3 Framework, with builds available for MacOS, Linux, and Windows.
- User Interface: Intuitive and responsive, custom-built for the fast-paced, demanding environment of clinical research.
- Functionalities: Equipped with data input forms, robust search and query capabilities, and specialized data visualization tools (coming soon) designed for clinical scientists.
- Data Policy: Adheres to a strict data policy ensuring that all data remains local, fortifying user trust and GDPR compliance.

3.2 Database

- Database Engine: SQLite
- Data Model: Utilizes the i2b2 Common Data Model (CDM) 16 star schema, optimized for the agile storage and retrieval of multimodal clinical data.

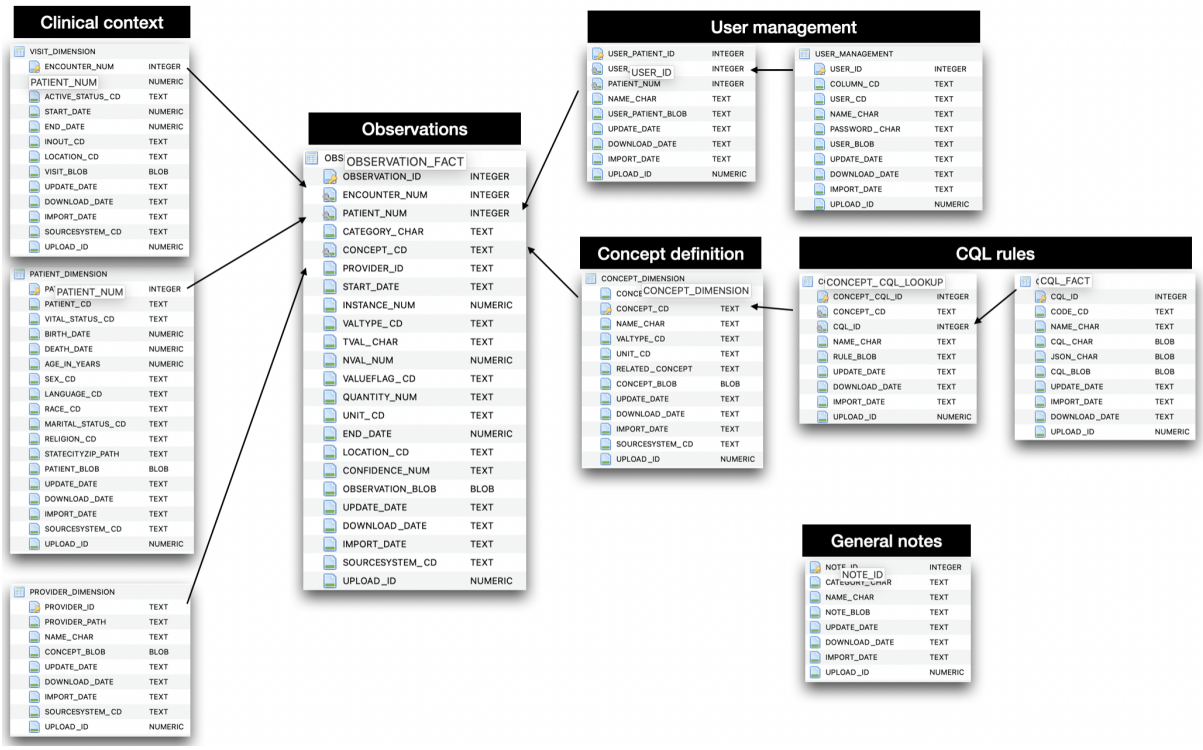


Fig. 1: Modified i2b2 Common Data Model (CDM) 16 star schema with additional tables for user management and CQL rules application.

CHAPTER

FOUR

4. TECHNICAL DETAILS:

Section	Description	Version	License	Website / Additional Info
Frontend				
VueJS 3 Framework		3.0.5	MIT	*VueJS Official Website <https://vuejs.org/>*
Quasar Framework		2.0.0	MIT	*Quasar Official Website <https://quasar.dev/>*
Libraries	crypto-js fs sqlite3	4.1.1 0.0.1 5.1.4		
Database				
SQLite		3.36.0	Public	*SQLite Official Website <https://www.sqlite.org/index.html>*
Tables	11 Tables			(view figure above)
Views	3 Views			cql_concept_list, patient_list, patient_observations
Triggers	3 Triggers			delete_concept_cql_lookup, delete_patient_cascade, delete_visite_cascade
Suggested Editor	DB Browser for SQLite	3.12.2	Public	*DB Browser for SQLite Official Website <https://sqlitebrowser.org/>*

CHAPTER

FIVE

5. KEY FEATURES:

Features	Description	Details
1. UI	lightweight frontend	VueJS 3, Quasar Framework
• multiplatform	Electron Framework	precompiled versions for MacOS, Linux, Windows
• adaptable	Easy customable	HTML/CSS based design
2. Database	SQLite DB	predefined tables, views, triggers, *see Appendix for details*
• lightweight	single file	no installation required
• fast	optimized queries	i2b2 CDM 16 star schema
• secure	user authentication	user management is realized via access rights to the database file, pure offline solution without internet connection required
5. Standardized concepts		redefined concepts based on LOINC, SNOMED/CT, ICD10, custom definitions
4. Data Input	via UI	customable input forms
• single observations	add observations	available
• set of observations	via schemes	predefined sets of Concepts and observations
• meta data		each observation is linked to a visit and a patient including additional meta data, timestamps, and user information AND is defined by a concept (e.g. LOINC) defining the type of observation
• CQL rules		CQL rules can be applied concepts and will be executed automatically when a new observation is added, *see Appendix for details*
• supported data types		all data types supported by SQLite (e.g. text, integer, float, blob) added support for Images and RAW data
5. Data Search		
• within the UI	searching for subject data and properties	customable search form with SQLite query available

5.1 Entering Data / Data Types

A major focus of this research database application/concept is the collection of different types of data. Therefore, this section will focus on this aspect.

5.1.1 Data Types and definition of schemes

The data type of a variable is defined by the CONCEPT_CD field in the CONCEPT_DIMENSION table (*please refer to the appendix of this section for further details*). The following principle data types are supported:

Data Type	Description
N	Coded for numeric data.
T	Coded for textual data.
D	Encoded for date types and follows the YYYY-MM-DD format.
R	Coded for raw data, to accommodate unprocessed or unformatted information directly from the data source. This type is used for variable images and other binary data.
F	Coded for findings, indicating whether a particular clinical feature is present, with options such as 'yes', 'no' and 'unknown'.
S	Coded for choices, often showing answers attached in the CONCEPT_PATH.
A	Coded for answers to choices ('S'), often showing answers attached in the CONCEPT_PATH

The following figures show the definition of a scheme using various data types:

Fig. 1: Screenshot from the user interface showing the definition of a schema with different data types.

Note: The different observations are associated with different data types. Note that there are different types of *MoCA*, one representing a *finding* (indicating that the patient has been tested for *MoCA*), the other representing the *numeric value* of the test.

LID indicates a LOINC-ID, which is a unique identifier for a particular LOINC concept.

5.2 Scheme definition

Note: *Schemes* represent a set of observations. Defining a scheme and consequently using the scheme for data input is a fast and efficient way to enter data. However, it is also possible to enter data without defining a scheme. This is useful for one-time data entry or for data that is not part of a scheme.

The *Schemes* can be managed using the UI in the *Settings / Schemes* tab. Administrative rights are required to manage the schemas.

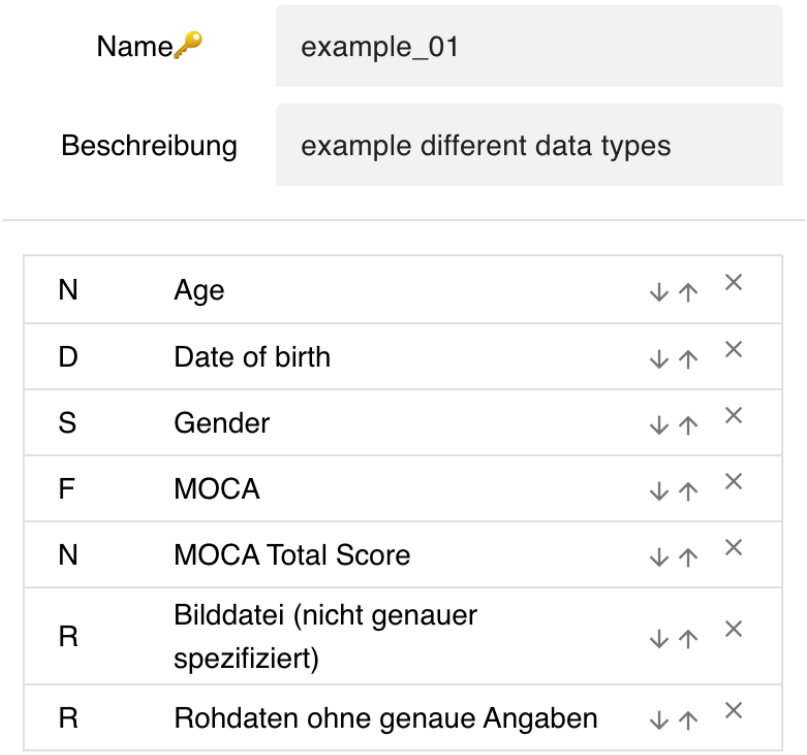


Fig. 2: Further details of the example scheme.

The scheme definition is stored within the database in the *CODE_LOOKUP* table.

The following image shows the scheme definition of the example scheme from the previous section:

5.3 Entering data using a scheme

To **input observations using a scheme**, a *patient* and a *visit* must be selected in the UI. By clicking on the *Add* button, the user can select a scheme from the list of available schemes.

The data is then stored in the database to the *OBSERVATION_FACT* table. The following images shows the data that was entered using the scheme from the previous section:

Note: Instead of using the column *OBSERVATION_FACT*, the view *patient_observations* is employed. This view translates *CONCEPT_CD* to *CONCEPT_NAME_CHAR* and *TVAL_CHAR* to *TVAL_RESOLVED*.

5.4 Concepts

Note: Further details can be found in the Appendix.

In our database design, the CONCEPT_DIMENSION plays a key role in defining the nature of observations derived from various sources such as LOINC or Snomed/CT. Each concept is identified by a unique CONCEPT_CD, which serves as a definitive code for the observed data. To further specify the type of data an observation contains, we introduce the VALTYPE_CD field.

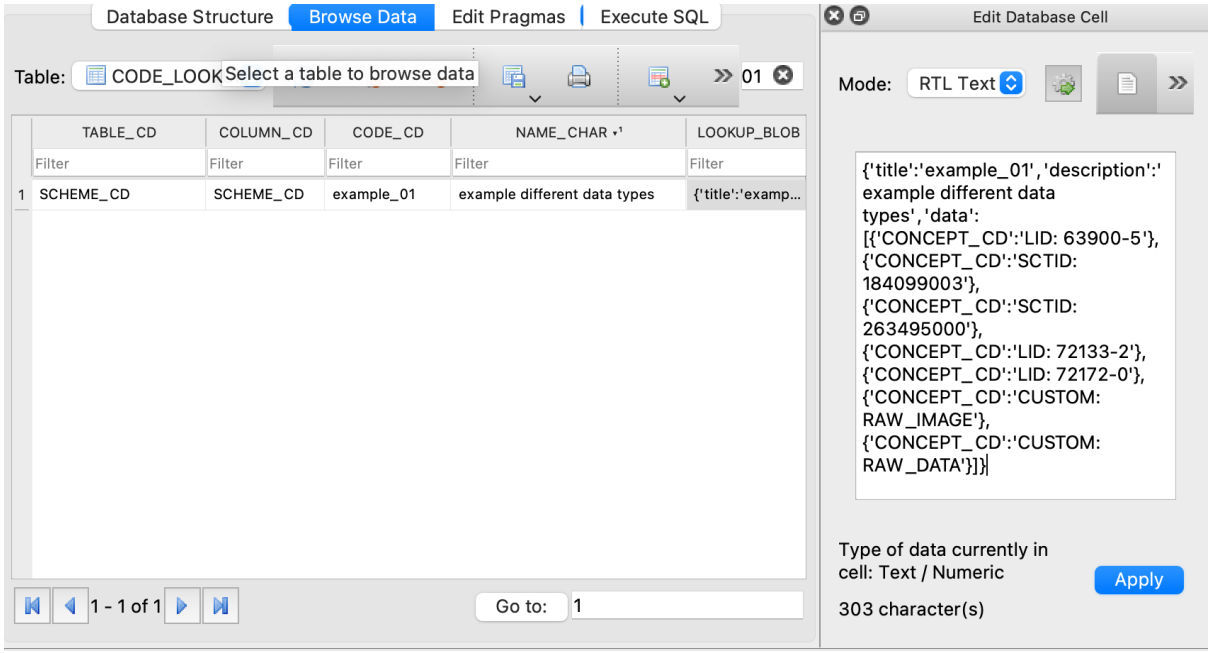


Fig. 3: Screenshot created using SQL DB Browser showing the scheme definition in the database.

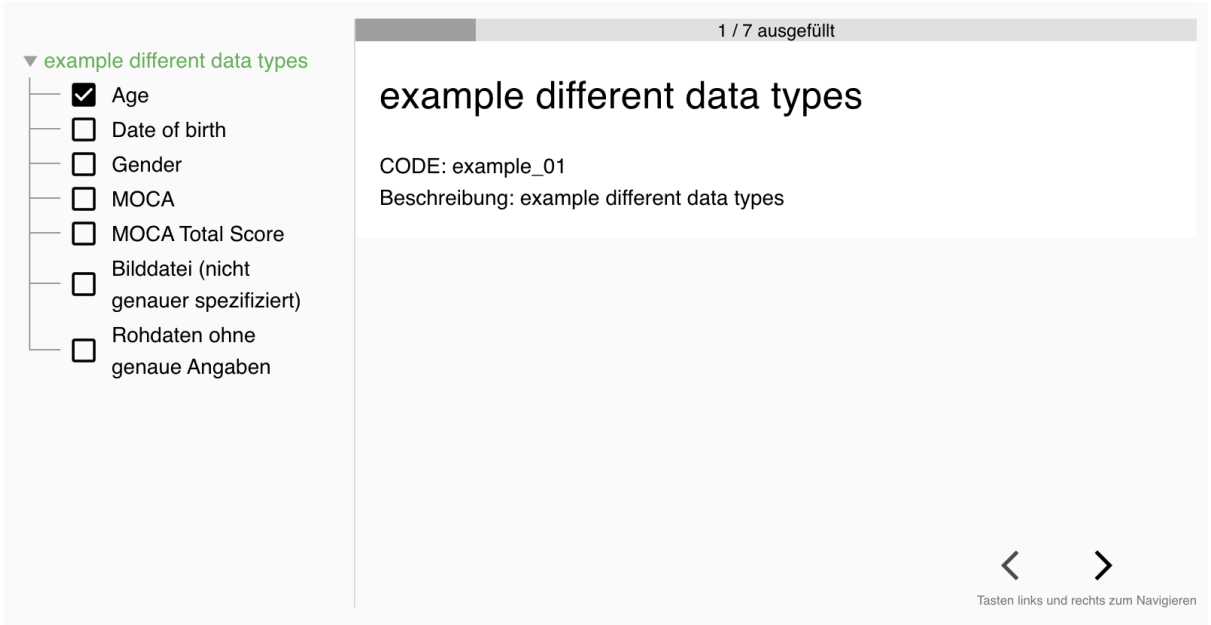


Fig. 4: Screenshot from the user interface showing a scheme.

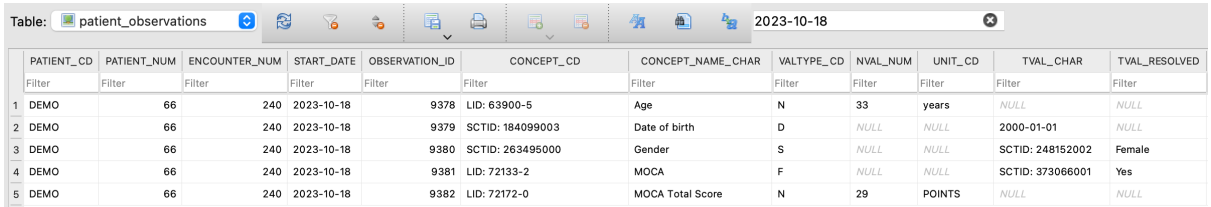


Fig. 5: Screenshots from the user interface display data entered according to the schema outlined in the previous section, as viewed through *SQL DB Browser*.

5.4.1 Overview

The following table provides an overview of the concepts that have been implemented.

Concept	Description	API	External Reference
LOINC	Logical Observation Identifiers		[LOINC Website](https://loinc.org)
SNOMED-CT	Systematized Nomenclature of Medicine	X	[SNOMED Website](https://www.snomed.org)
ICD10	International Classification of Diseases		[ICD10 Website](https://www.who.int/classifications/icd/en/)
CUSTOM	Customized Codes		N/A

5.4.2 Managing Concepts

Note: For SNOMED, we offer an integrated SNOMED-CT API that can be accessed directly from the user interface during concept editing.

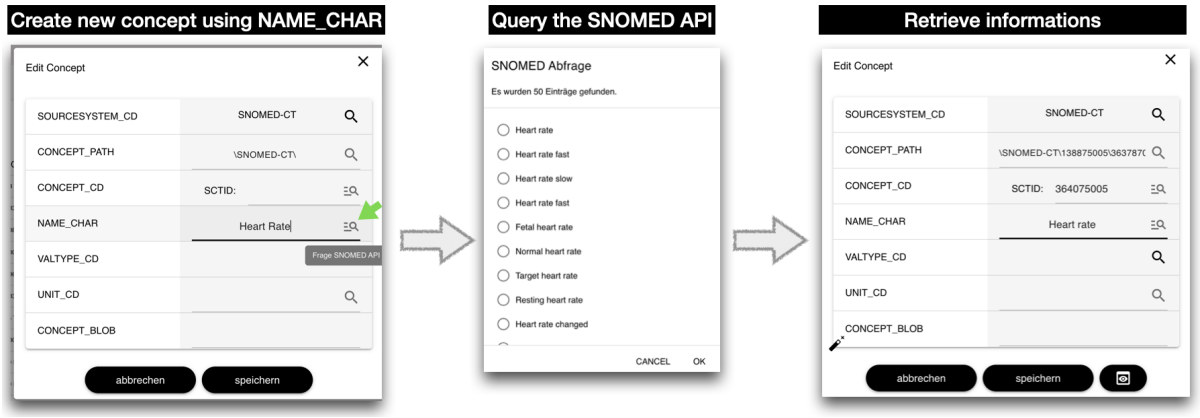


Fig. 6: When you select *SNOMED* as the *SOURCESYSTEM_CD*, an API query option becomes available, symbol is denoted by a green arrow. This query feature allows you to search for *SNOMED* concepts and obtain the corresponding *CONCEPT_CD* and *CONCEPT_PATH*, or to retrieve information associated with a specific *CONCEPT_CD*.

Warning: This marks the sole external connection the database currently utilizes. Only information related to concepts will be transmitted. The URLs for this functionality are hardcoded in the corresponding JavaScript file.

```
const url_full = 'https://browser.ihtsdotools.org/snowstorm/snomed-ct/browser/MAIN/
↪2023-02-28/concepts'
const url_short = 'https://browser.ihtsdotools.org/snowstorm/snomed-ct/MAIN/2023-
↪02-28/concepts'
const url_descriptions = 'https://browser.ihtsdotools.org/snowstorm/snomed-ct/
↪browser/MAIN/descriptions'
```

Note: Concepts are stored with the SQLite database in the table CONCEPT_DIMENSION. The table is created when the database is initialized.

The UI allows concepts to be managed (added, edited, deleted). In addition, concepts can be exported and imported from the UI using a JSON file.

5.4.3 Related Concepts

When using concepts of the VALTYPE_CD = 'S' (selection) answers are defined by the hierarchy of the concept in the CONCEPT_PATH.

Sometimes, similar answers should be provided for different concepts, i.e.

- NIHS Score Item: 4a. Left Arm (LID: 70190-4)
- NIHS Score Item: 4b. Right Arm (LID: 70967-5)

Therefore we introduced the concept of concept aliases or *related concepts*. This is illustrated in the following figure.

	CONCEPT_PATH	CONCEPT_CD	NAME_CHAR	VALTYPE_CD	UNIT_CD	RELATED_CONCEPT	CONCEPT_BLOB
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4	LID: 70190-4	Motorik Arme Links [NIHSS]	S	*NULL*	*NULL*	*NULL*
2	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4\LA\1	LID: LA18456-6	0	A	*NULL*	*NULL*	No drift
3	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4\LA\2	LID: LA18457-4	1	A	*NULL*	*NULL*	Drift
4	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4\LA\3	LID: LA18458-2	2	A	*NULL*	*NULL*	Some effort against gravity
5	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4\LA\4	LID: LA18459-0	3	A	*NULL*	*NULL*	No effort against gravity
6	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70190-4\LA\5	LID: LA18460-8	4	A	*NULL*	*NULL*	No movement
7	\LOINC\PANEL.SURVEY.NEURO\NIHSS\70182-1\70967-5	LID: 70967-5	Motorik Arm rechts [NIHSS]	S	*NULL*	LID: 70190-4	*NULL*

Fig. 7: The concept *LID: 70190-4* serves as the primary concept, whereas *LID: 70967-5* functions as an alias. These two are linked through the *RELATED_CONCEPT* column. Consequently, the alias inherits the same set of answers as the primary concept.

5.5 Export and Import of Data

The database supports the export of data in CSV format and HL7 JSON format. The CSV format is a simple text format that can be opened with any text editor or spreadsheet program. The HL7 JSON format is a standardized format for the exchange of clinical data. The database uses the HL7 JSON format to exchange data with other instances of the database or other systems.

Note: In the Appendix you will find exemplary CSV and HL7 JSON files.

CHAPTER

SIX

6. SECURITY AND COMPLIANCE:

In its intended use within the network of the University Hospital Jena, the database is protected by the following security measures - the database is only available within the local network of the hospital - the location of the database is shared only within the research study group - only authorised users have access to the database

The database is designed to be used in a secure environment. It is not recommended to use the database in an unsecured environment. The database is not designed for use in a public environment.

By default, the research database is designed to work with data that is not considered personal data under the GDPR. However, the database may be used to store personal data. In this case, the user is responsible for compliance with the GDPR. The database is not designed for use in a public environment.

CHAPTER

SEVEN

7. INTEGRATION & EXTENSIBILITY:

The UI front-end is a standalone application and can connect to any SQLite database file structured according to the template provided. The database file can be exchanged with other instances of the database or other systems using the HL7 JSON file standard.

There are currently no APIs available for direct interaction with the user interface. However, the database can be accessed directly via SQL queries using the suggested *DB Browser for SQLite* editor: https://sqlitebrowser.org.

7.1 Integration der SQLite DB in KNIME

To analyze the data directly from the database, the SQLite DB can be seamlessly integrated into KNIME 5.1 (https://www.knime.com).

Below is an example illustrating how to incorporate the data from the SQLite DB into KNIME.

Note: The SQL Statement for the *DB Query Reader* is stated below:

```
SELECT *, CAST(NVAL_NUM AS REAL) as NVAL_NUM_REAL
FROM patient_observations;
```

Note: The python script to transform the data is provided in the appendix.

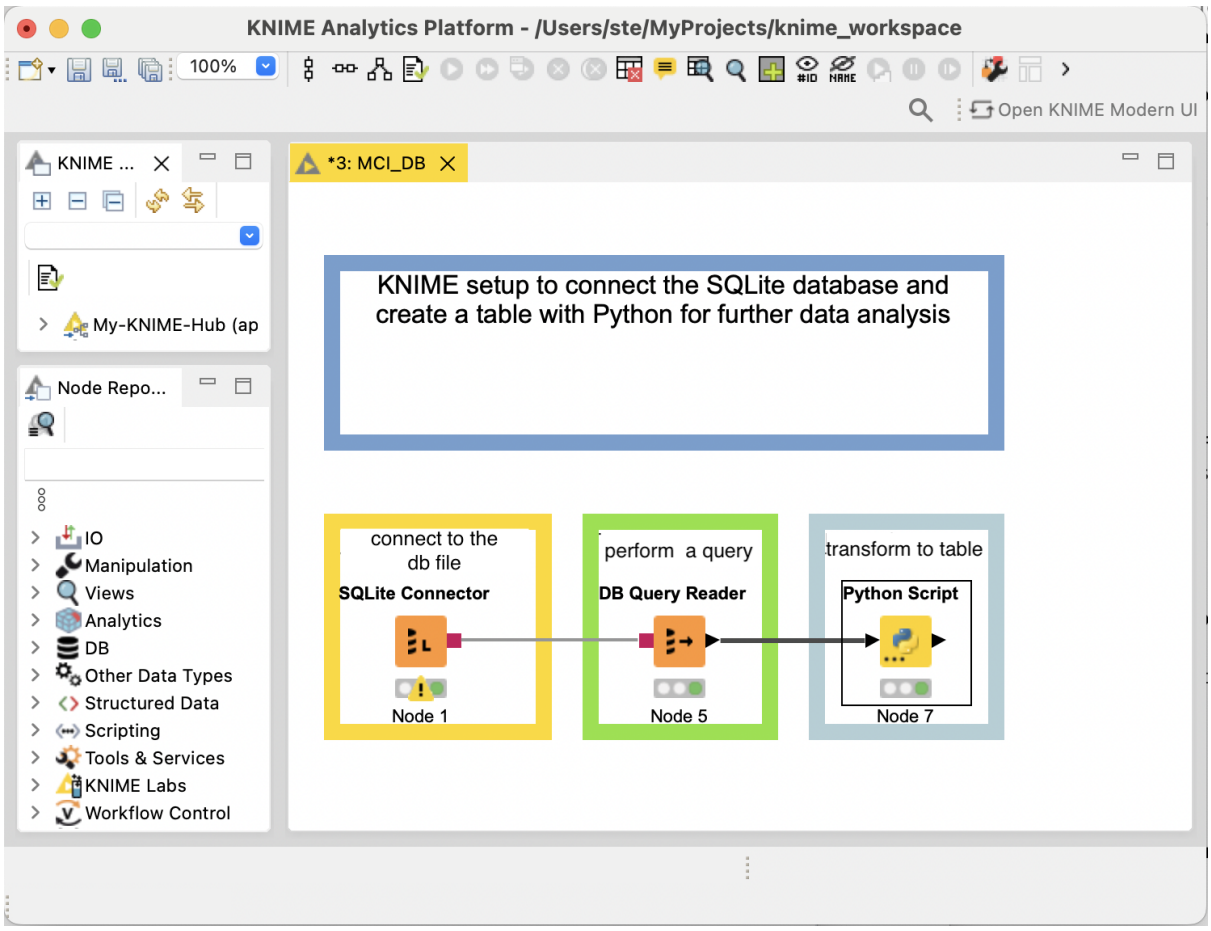
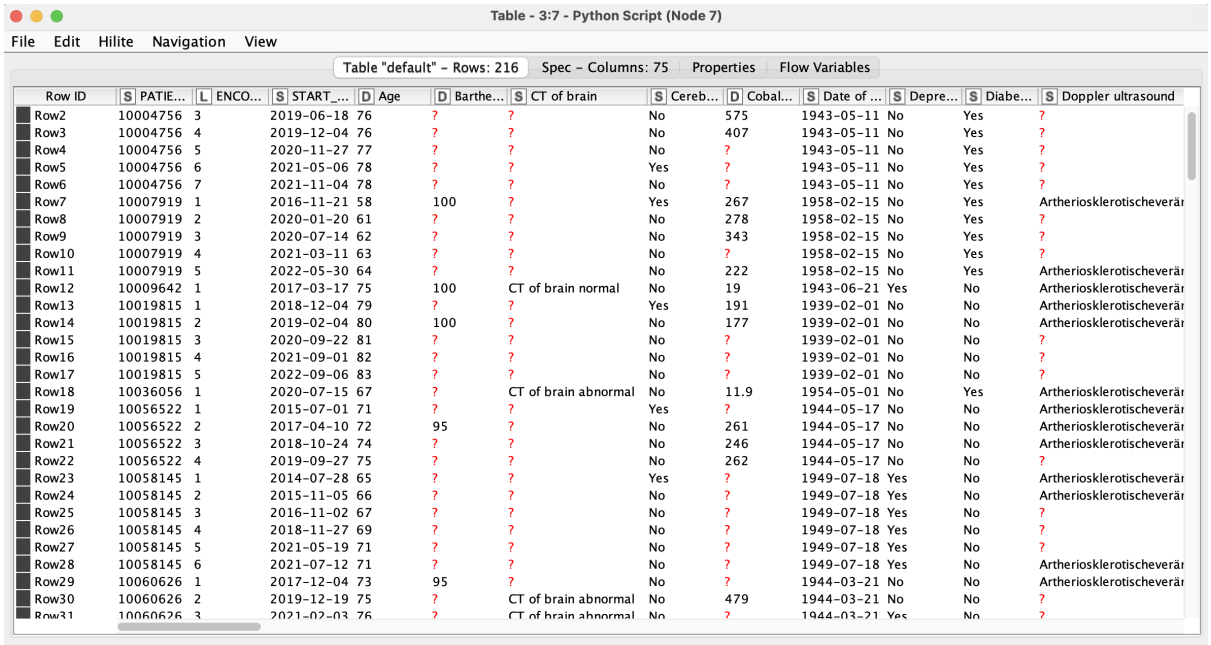


Fig. 1: Sample KNIME workspace to connect to the DB and transform the data into a tabular format suitable for further analysis.



Row ID	PATIE...	ENCO...	START_...	Age	Barthe...	CT of brain	Cereb...	Cobal...	Date of ...	Depre...	Diabe...	Doppler ultrasound
Row2	10004756	3	2019-06-18	76	?	?	No	575	1943-05-11	No	Yes	?
Row3	10004756	4	2019-12-04	76	?	?	No	407	1943-05-11	No	Yes	?
Row4	10004756	5	2020-11-27	77	?	?	No	?	1943-05-11	No	Yes	?
Row5	10004756	6	2021-05-06	78	?	?	Yes	?	1943-05-11	No	Yes	?
Row6	10004756	7	2021-11-04	78	?	?	No	?	1943-05-11	No	Yes	?
Row7	10007919	1	2016-11-21	58	100	?	Yes	267	1958-02-15	No	Yes	Arteriosklerotischeverär
Row8	10007919	2	2020-01-20	61	?	?	No	278	1958-02-15	No	Yes	?
Row9	10007919	3	2020-07-14	62	?	?	No	343	1958-02-15	No	Yes	?
Row10	10007919	4	2021-03-11	63	?	?	No	?	1958-02-15	No	Yes	?
Row11	10007919	5	2022-05-30	64	?	?	No	222	1958-02-15	No	Yes	Arteriosklerotischeverär
Row12	10009642	1	2017-03-17	75	100	CT of brain normal	No	19	1943-06-21	Yes	No	Arteriosklerotischeverär
Row13	10019815	1	2018-12-04	79	?	?	Yes	191	1939-02-01	No	No	Arteriosklerotischeverär
Row14	10019815	2	2019-02-04	80	100	?	No	177	1939-02-01	No	No	Arteriosklerotischeverär
Row15	10019815	3	2020-09-22	81	?	?	No	?	1939-02-01	No	No	?
Row16	10019815	4	2021-09-01	82	?	?	No	?	1939-02-01	No	No	?
Row17	10019815	5	2022-09-06	83	?	?	No	?	1939-02-01	No	No	?
Row18	10036056	1	2020-07-15	67	?	CT of brain abnormal	No	11.9	1954-05-01	No	Yes	Arteriosklerotischeverär
Row19	10056522	1	2015-07-01	71	?	?	Yes	?	1944-05-17	No	No	Arteriosklerotischeverär
Row20	10056522	2	2017-04-10	72	95	?	No	261	1944-05-17	No	No	Arteriosklerotischeverär
Row21	10056522	3	2018-10-24	74	?	?	No	246	1944-05-17	No	No	Arteriosklerotischeverär
Row22	10056522	4	2019-09-27	75	?	?	No	262	1944-05-17	No	No	?
Row23	10058145	1	2014-07-28	65	?	?	Yes	?	1949-07-18	Yes	No	Arteriosklerotischeverär
Row24	10058145	2	2015-11-05	66	?	?	No	?	1949-07-18	Yes	No	Arteriosklerotischeverär
Row25	10058145	3	2016-11-02	67	?	?	No	?	1949-07-18	Yes	No	?
Row26	10058145	4	2018-11-27	69	?	?	No	?	1949-07-18	Yes	No	?
Row27	10058145	5	2021-05-19	71	?	?	No	?	1949-07-18	Yes	No	?
Row28	10058145	6	2021-07-12	71	?	?	No	?	1949-07-18	Yes	No	Arteriosklerotischeverär
Row29	10060626	1	2017-12-04	73	95	?	No	?	1944-03-21	No	No	Arteriosklerotischeverär
Row30	10060626	2	2019-12-19	75	?	CT of brain abnormal	No	479	1944-03-21	No	No	?
Row31	10060626	3	2021-02-03	76	?	CT of brain abnormal	No	?	1944-03-21	Yes	No	?

Fig. 2: Screenshot from the output of a python script node, which is used to transform the data into a tabular format suitable for further analysis.

CHAPTER

EIGHT

8. TESTING & VALIDATION:

Testing Frameworks:

- **Unit Tests: using the Jest Framework (https://jestjs.io/) for testing**
  - data import and export
  - CRUD operations on the database
  - CQL rules application
- **Integration Tests: using the Cypress Framework (https://www.cypress.io/) for testing**
  - UI testing (e.g. data input, data search, data export)

CHAPTER

NINE

9. CONCLUSION & FUTURE WORK:

The Research Database is a simple yet powerful tool for storing all types of clinical data in a single place. It is designed to be easily extensible, allowing new data types and functionalities to be added. Data can also be exchanged with other database instances or other systems using a defined HL7 JSON file standard.

The status as is is a first version of the database. It is already in use at the University Hospital of Jena and will be further developed in the future.

The following features are planned for future releases:

- Data Visualisation
- Built-in image and RAW data viewer
- Enhanced security and compliance through database file encryption

CHAPTER

TEN

10. ACKNOWLEDGEMENTS:

The development of the database was supported by the following institutions - Department of Neurology, University Hospital Jena - IMSID - Institute for Medical Statistics, Informatics and Data Science, University Hospital Jena

The following persons contributed to the development of the database - PD Dr. Stefan Brodoehl - Anna Schweinar, cand. med.

CHAPTER

ELEVEN

11. APPENDIX:

Only Available in the HTML Version.