Max Planck Institute for Metabolism Research

Dr. Stefan Vollmar
Head of IT and Software Development

vollmar@sf.mpg.de
mobile: 0160-93874279
https://www.sf.mpg.de/it-group

»StudyDB« — Key Concepts

Maria Müller, Michael Sué & Stefan Vollmar

Abstract

We have presented a number of database applications over the past years (ScoreDB [1], GeneDB [2], TVADB [3]) and our new StudyDB inherits a lot of functionality from these previous Django development efforts.

However, StudyDB faces some additional challenges as it is intended for data collection in the context of translational human studies (empirical and prospective research, clinical trials) and »electronic« questionnaires (interfacing with and processing data from our LimeSurvey [4] server). The »clinical trials« part alone requires a significant amount of »quality assurance« (it seems you can get away with a lot of very bad technical choices as long as they are nicely documented; we tried a more sporting approach).

We would like to present our implementation of the following key concepts which might be of a more general interest and are not limited to database applications:

Single Source of Truth

StudyDB manages about 1000 parameters in about 30 tables. The scientists conducting the studies assembled JSON files (YAML or XML would also have worked in principle) for each table describing each parameter with its data type (they needed some help with this, of course), expected range of values and a comment including the units of measurements if applicable (all files are managed in one repository on our Github server), see Fig. 1.

- We use JSON-Schema [8] to formally validate these JSON files.
- We have written a Python program to generate the required Django (Python) code for the database models (Django's database abstraction layer) from the JSON files.
- The JSON descriptions allow for easy and efficient simulation of test data (with some help from the Faker library).
- We use those JSON files for parsing any input data (import is atomic and requires each data set to pass validation) and choosing the display format, see Fig. 2.
- Previously, clinicians used Excel spreadsheets for calculating some study parameters. This is not a good workflow (lack of audit trail, off-by-one-errors in Excel formulas, our general anti-Excel-bias) and we define »function«-type fields that present as »normal« database entries (which they are) — but they are re-calculated if any of their parameters is updated.
- The JSON files are used to create an automated »Electronic Case Report Form« (PDF via a LaTeX template), see Fig. 3.

Fig. 1 Examples of table descriptions. We needed to adapt the JSON structure from our initial draft — this way allows using JSON Schema validation for each field.

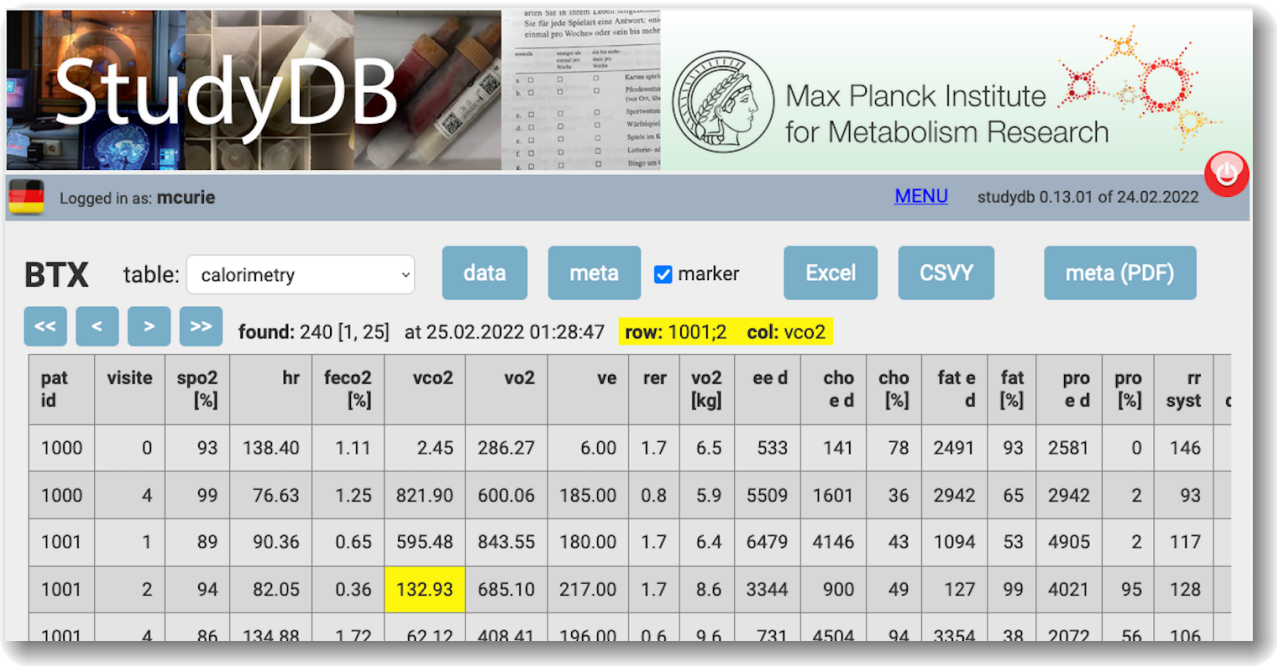


Fig. 2 Our generic table viewer that allows browsing all tables of all studies (provided the currently logged-in user has appropriate permissions) by efficiently accessing Django data at object level. Here shown in »marker mode« (the yellow elements) which also support fast keyboard navigation (and might later be used for online editing of individual cells). Scrolling is limited to the inner table area (top buttons and text remain static). Formatting is derived from the JSON configuration of each table.

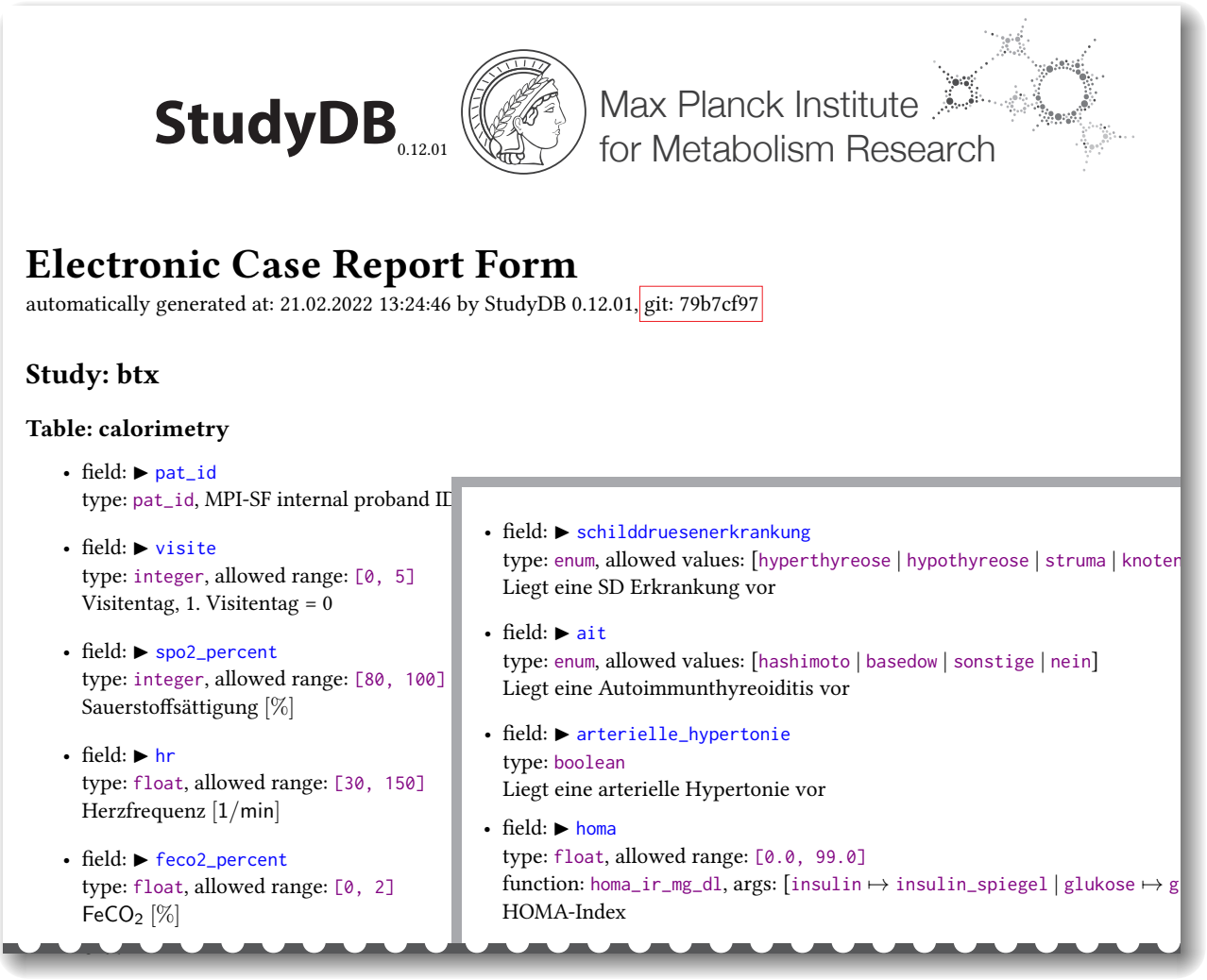


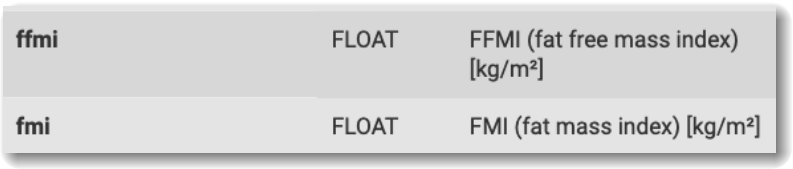
Fig. 3 Automatically generated eCRF report using LaTeX for PDF generation. Information extracted from the structured description (JSON) of the tables and used for validation and other purposes has been typeset in a different style. Of particular interest is the field marked »homa« which demonstrates an option to define formula-type results based on other parameters of one dataset. The report is being generated using all configuration files for all tables which are managed in one central git repository: the abbreviated hash (here marked red) in the report's head identifies the exact state of the repository.

Timestamping

Any data uploaded to StudyDB will be subjected to »timestamping« as described in RFC3161. We briefly considered using Bloxberg or some other blockchain technique but could see no advantages over using the lightweight and elegant »DFN Zeitstempeldienst« [6] for now: we create an upload/processing log with SHA256 hashes of all files involved. That report file will be hashed, the hash transferred to the external (and trusted) service and the digitally signed receipt (with timestamp) returned from the service will be archived. Future audits can be conducted offline and only require a suitable public key to verify the timestamp.

Generic Table Viewer

The large number of tables and parameters renders manual configuration of tables for viewing/browsing purposes impractical here. We had already written a generic table viewing class which allowed for easy Excel and CSVY [7] export and could harness this code for object-level database access — one example where profiling information from Django Debug Toolbar [10] was helpful to identify a bottleneck. Our table viewer uses KaTeX [5] to render tagged LaTeX parts of cell data, see Fig. 4.



ffmi	FLOAT	FFMI (fat free mass index) [kg/m²]
fmi	FLOAT	FMI (fat mass index) [kg/m²]

Fig. 4 Each table can be viewed in »meta« mode showing the information from the JSON config files for each field. We can tag parts of a comment text which is then rendered by KaTeX and thus enabling the full range of LaTeX math typesetting even for HTML views.

References

[1] DV-Treffen Göttingen 2019, »ScoreDB: Django-Anwendung im Labor mit Tablets und Barcode-Scanner«, https://wiki.init.mpg.de/share/Workshops/DVTreffen/36DVT?action=AttachFile&do=view&target=dv-treffen-2019-scoredb-v08p.pdf

[2] DV-Treffen Online 2020, »Vortrag/Live-Demo: Django-Datenbanken im Labor, Beispiel GeneDB«, https://wiki.init.mpg.de/share/Workshops/DVTreffen/37DVT?action=AttachFile&do=view&target=vollmar-dv-treffen-2020-genedb-v14.pdf

[3] DV-Treffen Online 2021, »Django-Anwendungen mit Archivierung«, https://wiki.init.mpg.de/share/Workshops/DVTreffen/38DVT?action=AttachFile&do=view&target=vortrag-dv-vollmar-2021-v16p.pdf

[4] LimeSurvey: OpenSource-Lösung für »elektronische Fragebögen«, https://www.limesurvey.org

[5] KaTeX, »The fastest math typesetting library for the web.«, https://katex.org

[6] Zeitstempeldienst (DFN), https://www.pki.dfn.de/zeitstempeldienst

[7] CSVY: yaml frontmatter for csv file format, https://csvy.org

[8] JSON Schema, »JSON Schema is a vocabulary that allows you to annotate and validate JSON documents.«, https://json-schema.org

[9] Faker: »Faker is a Python package that generates fake data for you«, https://faker.readthedocs.io

[10] Django Debug Toolbar: »The Django Debug Toolbar is a configurable set of panels that display various debug information about the current request/response and when clicked, display more details about the panel's content.«, https://django-debug-toolbar.readthedocs.io