Innovative form generator for recording complex support programmes

Alexander Aue [1], Andrea Ackermann[1] and Norbert Röder [1]

Abstract: In an effort to incentivize environmental protection actions among farmers, different administrations offer various support programmes. These programmes differ between states, change over time and are published in non-machine-readable formats. Since the lack of a nationwide database hampers the comprehensive evaluation of these programmes, we propose a flexible form generator for standardizing the programme descriptions. Our tool meets the needs of researchers as it facilitates the transformation of text-based definitions to a well-structured relational database with its graphical questionnaire user interface. Redundancies in data input are avoided by allowing the collected data of a support program to be inherited by another and to extend it by additional properties or overwriting existing ones. Furthermore, it accommodates complex data collection needs, supporting limitless sub-questions and customized input fields. The resulting hierarchical data is automatically stored in a relational database, ensuring a user-friendly experience both for quantitative researchers and data analysts. This innovative approach enhances the evaluation of e.g. environmental protection programmes.

Keywords: form generator, form builder, data collection, unstructured data, data inheritance, hierarchical data, relational data

1 Introduction

In order to provide farmers with an incentive to conduct more environmentally friendly farming practices, various administrations offer different support programmes. In the European Union, the programmes funded by the Common Agricultural Policy are of particular relevance [Eu22]. In Germany, the overwhelming majority of the support programmes is designed and administered at regional level [BM23], with the details published in natural language in paper form or as PDF files. In this form, however, the information is not machine-readable. The absence of a nationwide database containing these programmes inhibits the ability to comprehensively characterize, evaluate, and assess them across different funding periods and federal states. Therefore, the need for a nationwide database and a corresponding data collection form becomes evident.

In order to ensure data quality and consistency in a data collection form, it is advisable to avoid free-text inputs. Instead, predefined input methods should be used. This can be

[1] Thünen-Institute of Rural Studies, Bundesallee 64, 38116 Braunschweig, alexander.aue@thuenen.de, https://orcid.org/0009-0001-8683-3630; andrea.ackermann@thuenen.de; norbert.roeder@thuenen.de, https://orcid.org/0000-0002-2491-2624

achieved through techniques like single-choice and multiple-choice questions, radio buttons, checkboxes, date pickers and numeric inputs. Auto-suggestions can also guide users in providing structured data. By implementing these strategies, data collection becomes more organized, making analysis and management more straightforward while minimizing the risk of inconsistent or inaccurate data.

Existing form software available on the market, like Google Forms or SurveyJS, is typically designed to deliver user-friendly interfaces for standard top-level questions. They can significantly expedite the process of configuring a form compared to developing one from the ground up with the help of software developers. Another option is the Open Data Kit, which allows forms to be created by specifying the form structure within a spreadsheet, allowing greater customisation [Ha10]. However, the intricate nature of support programs cannot be effectively characterized by existing tools: For instance, support measures can extend upon other existing measures by introducing additional properties or modifying existing ones, so called top-ups. In such cases, custom solutions are necessary to accurately capture the details and interdependencies of these measures, which goes beyond the capabilities of readily available form-building tools. Furthermore, existing form builders often fall short in supporting sub-questions and detailed answer customizations. While some form builders do provide the option to present an additional panel or page when a particular answer is selected, these panels or pages are usually limited to organizing questions at the top level.

Consider a scenario where a multiple-choice question includes the same sub-question for every answer option. In a conventional form builder, developers would have to create a separate section for each set of sub-questions and then duplicate these sections for every top-level answer [Sh23; Cr23]. This duplication necessitates the use of unique identifiers to distinguish among them. On the user's end, keeping track of which section corresponds to which answer becomes a cumbersome task, even when dealing with just two levels of questions and answers. For a form that goes beyond two levels and includes three or more levels of nested questions, the process of creating and managing numerous sections representing nested questions becomes increasingly impractical for both developers and users. Matching sections to specific answers grows more challenging as complexity rises.

When handling these multiple nested layers of form fields and predefined values, the question arises about how to store that data. While a hierarchical database format like JavaScript Object Notation (JSON) or a NoSQL database like MongoDB may seem like an option, it results in semi-structured data that is challenging to analyse. A relational database presents a more suitable solution as they are widely adopted in land use research and therefore familiar to users who want to analyse the data.

Since support programmes depend on evolving political goals and vary over time, it becomes necessary to add fields to the form. An effective strategy is to dynamically generate forms based on an easily maintainable configuration, allowing advanced users to define fields and predefined values. As this configuration is then stored in the database, every user can get the updated configurations from the database directly, ensuring that the

form can easily adapt to evolving data collection needs without requiring constant developer intervention and program compilations.

1.1 Problem Statement

To address the need for a nationwide database containing support programmes, enabling comprehensive characterization, evaluation, and assessment across different funding periods and federal states, the development of a flexible and easily adaptable form generator is imperative. This form generator should fulfil the data collection requirements for support programmes while maintaining the user-friendliness of a simple relational database for subsequent analysis.

The strategy to achieve this goal is twofold:

1. We present a relational database model capable of storing hierarchical data for multiple-choice form fields and predefined values, with an unlimited number of nested multiple-choice form fields. This model also allows filled-out forms to be inherited by other filled-out forms, enabling the extension of existing forms with additional properties or manipulation of existing ones.
2. We introduce a form application that consumes a simple JSON configuration of the survey structure, specifying the order and types of form fields, along with conditional logic to show or hide certain fields and predefined values. This application stores form data in the relational database and offers features to duplicate existing filled-out forms, inherit them, extend them with additional properties, or overwrite existing properties.

2 Methods

2.1 Relational database model for support programmes

The relational database model used for storing the collected data is depicted in Fig. 1. This model was designed not only for support programmes but can also be applied for surveys in general. The *survey_type* table stores the form configuration as JSON in the *structure* column. The configuration for the data collection form for support programmes is represented by a single tuple in the *survey_type* table. However, the table can be used to store different versions of the form or configurations for forms with entirely different purposes. The *survey* table stores general data about the surveys, including the title, the user who created it, and the creation date, as well as the corresponding form configuration linked via the relationship with the *survey_type* table.

The *choice_question_answers* table contains the collected data for multiple-choice questions. This table links to the *survey* table to track which survey the answer belongs to.

The hierarchical structure enabling sub-questions and sub-sub-questions is established through the fact that tuples in *choice_question_answer* can link to other tuples in the same table via the relationship between *parent_question_id* and the primary key *id*.

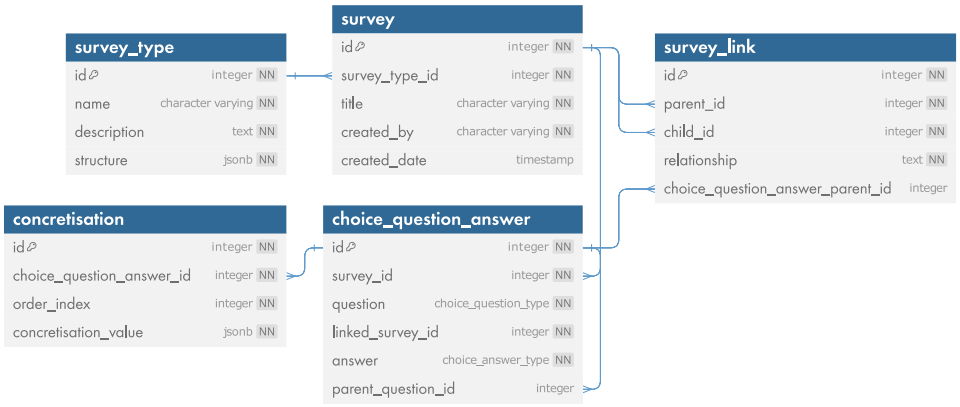

Fig. 1: Relational database model for surveys, including those for support programmes

The *concretization* table allows adding various types of details to each answer, linked through a relationship with the *choice_question_answer* table. The form builder supports custom input types tailored for support programmes, including time spans, numeric inputs with a selection of common agricultural units, and free text inputs for other purposes. New types of concretizations can be easily added because the *concretization_value* is flexible as it is stored as JSON. Finally, the *survey_link* table allows surveys to inherit from one another and decide which properties to inherit and which to omit.

2.2 Form Builder

An excerpt of the form for collecting support programmes is depicted in Fig. 2. For the purpose of this paper, the texts were translated into English. It displays the top-level questions on the left, with the question ‘Fertilization’ currently selected, and it presents the pre-set answers on the right.

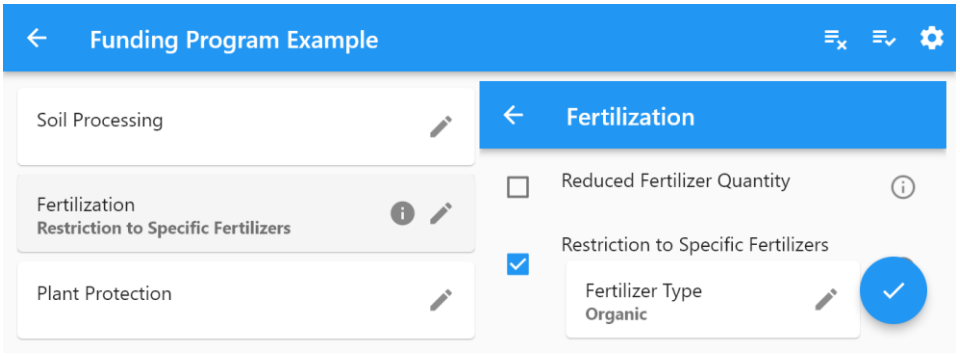

Fig. 2: Top-level questions as displayed in the generated form

The answer ‘Restriction to Specific Fertilizers’ allows for more details with the question ‘Fertilizer Type’. Clicking on that question opens the sub-question in the right panel, as shown in Fig. 3 on the left. Similarly, the answer ‘Organic Fertilizer’ can be further specified with the question ‘Organic Fertilizer Type’. Clicking on this question opens the sub-sub-question, depicted in Fig. 3 on the right.

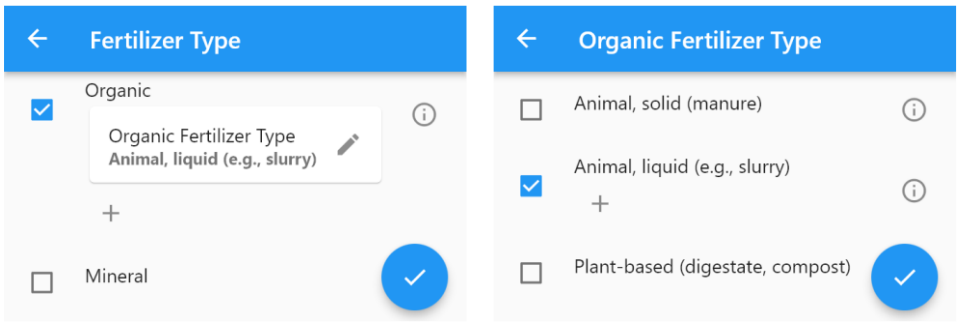

Fig. 3: Second level question (left) and third level question (right)

A simplified excerpt of the configuration for this example, showing only the first two levels and omitting IDs needed for the database, is shown in Listing 1.

```
{ "type": "MultipleChoiceQuestion",
  "title": "Fertilization",
  "choices": [
    { "title": "Reduced Fertilizer Quantity" },
    { "title": "Restriction to Specific Fertilizers",
      "subQuestions": [
        { "type": "MultipleChoiceQuestion",
          "title": "Fertilizer Type",
          "choices": [ { "title": "Organic" }
```

Listing 1: Simplified version of the configuration for the example shown in Fig. 2

3 Results

With the presented form generator, a form structure for support programmes was created, consisting of an extensive set of 47 single-choice and multiple-choice questions, along with 366 pre-set values. Using the generated form, over 300 agri-environmental schemes implemented by the German federal states in the CAP support period 2015-2022 were recorded and stored in a standardized database. First relevant queries were tested, such as compiling a complete systematic overview of existing measures.

4 Conclusion and Outlook

We have developed and applied a form builder with the capacity to generate data collection forms tailored for support programmes. Throughout the data collection process, new questions and predefined values had to be added frequently. Importantly, their incorporation into the existing form could be achieved without any interruptions in data entry or the intervention of a software developer. The resulting data base is currently being extended by adding remaining schemes from the period 2015-2022 and more importantly by adding the schemes for the time before 2015 and after 2022. This database will be primarily used for the general evaluation of CAP programmes, but also for research on the effectiveness and efficiency of support schemes for the promotion of biotic and abiotic resources.

Bibliography

[BM23] BMEL: CAP-Strategic Plan for the Federal Republic of Germany. https://www.bmel.de/SharedDocs/Downloads/DE/_Landwirtschaft/EU-Agrarpolitik-Foerderung/gap-strategieplan-version-2-0.pdf?__blob=publicationFile&v=5, accessed 27 Oct 2023.

[Cr23] Create a Multi-Page Survey | SurveyJS Form Libraries. https://surveyjs.io/form-library/documentation/design-survey/create-a-multi-page-survey#configure-page-visibility, accessed 26 Oct 2023.

[Eu22] European Court of Auditors: Biodiversity on farmland: CAP contribution has not halted the decline. https://op.europa.eu/webpub/eca/special-reports/biodiversity-13-2020/en/, accessed 14 Dec 2023.

[Ha10] Hartung, C. et al.: Open data kit. In (Unwin, T. Ed.): Proceedings of the 4th ACMIEEE International Conference on Information and Communication Technologies and Development. ACM, New York, NY, pp. 1–12, 2010.

[Sh23] Show questions based on answers - Google Docs Editors Help. https://support.google.com/docs/answer/141062, accessed 26 Oct 2023.