MSc thesis in Geomatics

Formalizing land indicators for SDGs: Implementation and evaluation using international standards

Mengying Chen

2024

TU Delft

MSc thesis in Geomatics

Formalizing land indicators for SDGs: Implementation and evaluation using international standards

Mengying Chen

October 2024

A thesis submitted to the Delft University of Technology in partial fulfilment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

Geo-Database Management Centre
Delft University of Technology

Supervisors:	Eftychia Kalogianni
	Professor dr. Peter van Oosterom
Co-reader:	Assistant Professor dr. Javier Morales Guarin

Abstract

This thesis explores the integration of the Land Administration Domain Model (LADM) with Sustainable Development Goal (SDG) indicator 1.4.2, which focuses on land tenure security and the availability of legal documentation. The primary objective is to design and implement a dynamic database system that simulates real-world land administration changes over time, incorporating the complexities of land rights transfers, party relationships, and spatial units. The research leverages PostgreSQL and PostGIS to create a flexible and scalable system capable of managing temporal data and generating SDG 1.4.2 reports.

Through the use of multiple constraints and custom functions, the system ensures data integrity while dynamically tracking changes in land tenure and ownership. Testing was conducted using simulated datasets across three years, which modeled evolving land rights, population changes, and administrative updates. The results were validated by comparing manual calculations with the system's automated outputs, demonstrating the accuracy and reliability of the approach.

The findings highlight the potential of using LADM in SDG indicator calculations and demonstrate the system's ability to handle complex, multi-dimensional land administration scenarios. However, limitations such as the use of artificial data and a focus solely on SDG 1.4.2 suggest that further work is needed to validate the model with real-world data and expand its application to other SDG indicators and land-related policies. This thesis contributes to the ongoing efforts to modernize land administration systems and provides a scalable solution for tracking land tenure security in line with global sustainability goals.

Acknowledgements

I would like to express my heartfelt gratitude to Peter and Eftychia. Collaborating with and being guided by them has truly been the luckiest part of my journey studying abroad. Our work on SDGs and LADM began over a year ago during the HPM project, when I didn't even know what LADM was. Faced with lengthy ISO documents full of specialized terminology, I felt completely lost. However, Peter and Eftychia were always patient and supportive, answering my questions during every meeting without ever making me feel that my queries were silly. My learning attitude and habits have been reshaped significantly through working with them. I deeply admire Peter's research approach—his passion for LADM is truly remarkable! His mastery and understanding of knowledge are unparalleled. No matter what ideas I brought forward, Peter always supported me and provided valuable suggestions.

With the encouragement of Peter and Eftychia, I had the opportunity to present at international conferences, publish articles in journals, and exchange ideas with other scholars. Through these experiences, I discovered my true passion for research. I realized how much I enjoy exploring new knowledge, sharing my thoughts with others, and presenting my work. I am deeply thankful to Peter for providing me with these opportunities, which allowed me to uncover my true interests.

The journey of writing this thesis was not without its challenges. I faced several hurdles, from changing my research topic to unexpected withdrawal of data from the provider, as well as balancing this thesis with the demands of my second master's degree. The pressure was overwhelming at times, but Peter and Eftychia consistently supported me. They helped me identify my priorities and find ways to maximize my time and energy efficiently.

I also feel extremely fortunate to have had the opportunity to study at TU Delft. The friendships I formed, the professors I learned from—I will deeply miss this period of my life. I sincerely hope to return to TU Delft in the future.

Contents

List of Figures

List of Tables

Acronyms

1. Introduction

1.1. Motivation

Land administration plays a crucial role in achieving the Sustainable Development Goals (SDGs). Effective land administration provides a framework that enhances agricultural sustainability and socio-economic growth by ensuring the clarity and protection of land rights, thereby reducing land disputes and improving agricultural productivity [Williamson et al., 2010]. Especially in areas facing environmental change and climatic pressures, as demonstrated by sustainable agricultural management practices in Central African Congo Basin countries by [Molua, 2014]. Moreover, sustainability in the land sector is essential to the successful implementation of multiple SDGs, especially at the national level, where land administration policies and actions play a key role in the realisation of these goals [Gao and Bryan, 2017]. Land administration provides the necessary technical, institutional and organisational support through capacity building in forest management [Carlson et al., 2015; Katila et al., 2020], agriculture and land restoration [Chigbu, 2023], which provides a global sustainable development a solid foundation [Bloomfield et al., 2018]. Through effective land use planning and management, land administration can also contribute to reducing environmental degradation, protecting ecosystems, and advancing the environmental sustainability objectives of the SDGs.

Monitoring the SDGs is a critical component in assessing global progress toward these goals. It not only facilitates the evaluation of national advancements in sustainable development but also identifies gaps and challenges specific to certain targets and indicators, thereby providing policymakers with scientific evidence to optimize resource allocation. However, globally, the monitoring of SDG progress faces a number of challenges, such as the lack and inconsistency of data [Tuholske et al., 2021], the gap between localisation and globalisation indicators [Chen et al., 2020], and the lack of harmonised monitoring tools. These challenges make it difficult to accurately assess SDGs progress at the global and local levels. Formalized and standardised indicators present a robust solution to these issues [Unger et al., 2021]. Firstly, it ensures consistency and clarity, reducing ambiguity in indicator definitions and calculations, and facilitating clear communication among stakeholders. Secondly, it enhances comparability, allowing for consistent comparison across different regions and time periods. Thirdly, it improves efficiency, enabling automated data processing and indicator calculation, thus saving time and resources. Finally, it ensures accuracy, providing a more precise representation of progress towards SDG targets.

In this context, the ISO 19152 Land Administration Domain Model (LADM) [**?**] offers a comprehensive framework for land administration that can significantly enhance the measurement and monitoring of SDGs. Its key components—such as LA_Party, LA_SpatialUnit, and LA_RRR—are widely used in various aspects of land rights, land use and resource management, as shown in Fig.1.1. In the formalisation and standardisation of SDG indicators, the LADM helps governments and international organisations maintain consistency

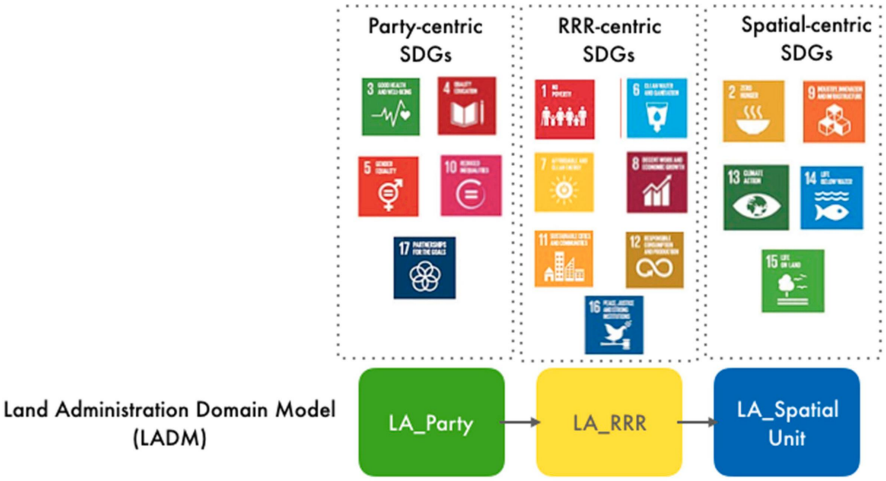


Figure 1.1.: LADM as a base to support the SDGs, adapted from [Unger et al., 2021]

in the collection and analysis of land-related data by providing consistent definitions and structures.

1.2. Research objective

The primary objective of this research is to explore the specific ways in which the ISO 19152 LADM can be implemented to support the formalization and standardization of land-related indicators within the SDGs. By validating whether the LADM framework can optimize the calculation and representation of these indicators, the study seeks to ensure their accuracy, consistency, and applicability across diverse regions and timeframes. The research focuses on how LADM, through its standardized structure—particularly in terms of data mapping, indicator calculation, and result comparison—can effectively support the formalization and standardization of SDG 1.4.2.

1.2.1. Research question

The main research question guiding this study is: "How can the ISO 19152 Land Administration Domain Model (LADM) be implemented to support the formalization and standardization of land-related Sustainable Development Goals (SDGs)?" This question will be explored by analyzing the alignment between LADM's standardized structure and the requirements for accurate SDG indicator monitoring, particularly in terms of data mapping, indicator calculation, and results comparison with existing reports.

To support the main research question, the following subquestions are proposed:

1. How can the conceptual model for SDG 1.4.2 be developed based on its metadata?
2. How can the conceptual model for SDG 1.4.2 be effectively translated into a physical database implementation?

3. What added value does this approach bring to the overall monitoring and evaluation of SDG indicators?

1.2.2. Scope

This research focuses on SDG indicator 1.4.2, which is particularly suitable for demonstrating the applicability of the LADM due to its two sub-indicators. The first sub-indicator, related to legally recognized documentation, can be fully expressed using the LADM framework, while the second sub-indicator, related to the perceived security of land tenure, requires an external class for support. By focusing on this indicator, the study examines the ability of the LADM to handle both legal and perceptual aspects of land tenure across a range of scenarios, providing insights into its broader applicability in other land-related SDG indicators.

Given the sensitivity of cadastral data, the study utilizes synthetic data, generated to model general cases without specific or exceptional characteristics. This approach ensures that the findings remain relevant across diverse geographic contexts while avoiding the use of sensitive or confidential data. The process of data generation and its application is detailed in later chapters, ensuring transparency and replicability in the modeling and analysis stages.

1.3. Thesis outline

The thesis is organized as follows:

Chapter 2 provides a comprehensive review of the background and related work in the field of land administration and sustainable development. It explores the foundational concepts of the LADM, its applications in various land administration processes, and its role in the formalization of SDGs indicators. The chapter also discusses previous studies that have implemented LADM for land management and highlights the research gaps that this thesis aims to address.

Chapter 3 details the methodology used in the research. It describes the transition from the conceptual model of LADM to a physical data model implemented in PostgreSQL. It explains the structure and key components of the physical data model, including the use of custom data types and standardized code lists. The chapter also covers the implementation of specific functions and triggers needed to ensure data consistency and facilitate indicator calculation. This section concludes with the process of mapping synthetic data to the LADM model and describes the calculation methods used for SDG indicator 1.4.2.

Chapter 4 presents the results of applying the LADM-based framework to SDG 1.4.2. It discusses the testing dataset, designed to simulate dynamic changes in land tenure, ownership, and administrative updates over time. The results of the automated system calculations are compared with manual computations to validate accuracy. The chapter also provides a detailed analysis of the system's performance in handling different temporal and spatial scenarios and discusses the reliability and accuracy of the approach.

Chapter 5 summarizes the findings of the research, highlighting the contributions made to the field of land administration and the monitoring of SDG indicators. It discusses the potential of the LADM framework for enhancing the formalization and standardization of

land-related indicators. The chapter also identifies limitations encountered during the research, such as the reliance on synthetic data, and proposes directions for future work, including the application of the model to real-world data and additional SDG indicators.

2. Background

2.1. Sustainable Development Goals

The SDGs were adopted by the United Nations in 2015 as a global development agenda aimed at achieving balanced progress in economic, social, and environmental dimensions by 2030 [Cf, 2015]. As shown in Fig 2.1, the SDGs consist of 17 goals and 169 targets, addressing a wide range of issues such as poverty eradication, hunger elimination, health and well-being promotion, quality education, gender equality, access to clean water and sanitation, economic growth, reducing inequalities, and climate action [Morton et al., 2017]. These goals reflect the core aspirations of global sustainable development, urging not only national governments but also various sectors of society to contribute to a comprehensive transformation across social, economic, and environmental domains.

Compared to their predecessor—the Millennium Development Goals (MDGs), which primarily targeted developing countries—the SDGs are more broadly inclusive and clearly defined [Sachs, 2012]. The SDGs were introduced at the 2012 United Nations Conference on Sustainable Development, where they received unanimous support from all 193 UN member states, as well as a wide array of non-governmental organizations. Unlike the MDGs, the SDGs emphasize universal applicability, addressing challenges faced by countries at all levels of development.

2.1.1. SDGs Indicators

To effectively monitor and evaluate the progress of the Sustainable Development Goals, a comprehensive indicator framework was established. These SDG indicators serve as mea-

Figure 2.1.: The 17 Sustainable Development Goals

surable benchmarks for governments, international organizations, and other stakeholders to track progress on the 17 overarching goals and 169 specific targets. Each indicator quantifies progress toward addressing the social, economic, and environmental challenges outlined in the SDGs, ensuring a data-driven approach to sustainable development [Hák et al., 2016].

As of 2024, there are 248 indicators (231 unique ones), categorized into three tiers based on the development of methodologies and the availability of global data [Global SDG Indicators, 2024]. Tier I indicators have established methodologies and are regularly reported by countries, Tier II indicators have methodologies but lack comprehensive data, while Tier III indicators initially lacked standardized methodologies but have been fully addressed, with no Tier III indicators remaining as of March 2024. The indicators are not unchangeable [Kim, 2023; Diaz-Sarachaga et al., 2018]. The Inter-agency and Expert Group on SDG Indicators (IAEG-SDGs) is responsible for developing and annually reviewing these indicators, determining if replacements, revisions, deletions, or additional indicators are needed [Lyytimäki, 2019]. National and local governments use these indicators to assess their progress in sustainable development and report their findings through voluntary national and local reviews [Schmidt-Traub et al., 2017].

Each SDG indicator comes with a metadata file, which serves as a critical document for understanding the indicator. These metadata files include key sections such as ”Indicator Information,” ”Data Reporter,” ”Definition, Concepts, and Classifications,” ”Data Source Type and Collection Method,” ”Other Methodological Considerations,” ”Data Availability and Disaggregation,” ”Comparability/Deviation from International Standards,” and ”References and Documentation.” Specifically:

- **Indicator Information** outlines the update date of the indicator, corresponding SDG goals, targets, and any related indicators.
- **Data Reporter** identifies the organizations responsible for reporting and monitoring the indicator.
- **Definition, Concepts, and Classifications** is one of the most important sections, as it defines key terms and measurement criteria, explaining how the indicator is constructed, its scope, and any specific classifications used.
- **Data Source Type and Collection Method** describes the types of data sources (e.g., surveys, censuses) and methods used for collecting the data, specifying any involved administrative or national statistical systems.
- **Other Methodological Considerations** addresses key rationales, limitations, and includes specific formulas used in the indicator’s computation.
- **Data Availability and Disaggregation** covers data availability, timeliness, and suggestions on how to disaggregate the data (e.g., by gender or region), providing guidance for reporting classifications.
- **Comparability/Deviation from International Standards** evaluates any discrepancies between national and international reporting standards.
- **References and Documentation** lists key resources, methodologies, and international frameworks guiding the implementation of the indicator.

SDG metadata files are crucial for guiding data collection and ensuring consistent application of methodologies across diverse contexts. It's important to note that the compilation of SDG indicator metadata files involves various entities, leading to differences in detail and update frequency.

2.1.2. SDGs Related to Land Administration

As shown in Figure 2.2, land administration plays a critical role in achieving multiple SDGs. It directly impacts economic growth, social equity, and environmental sustainability. Effective land administration ensures secure property rights, promotes responsible land use, and facilitates efficient land markets, all of which are vital for economic growth. Additionally, it supports environmental sustainability by fostering land-use practices aligned with ecological conservation objectives. Through good governance and spatial data infrastructure (SDI), land administration strengthens decision-making processes, and ensures that development policies are inclusive, transparent and equitable, which is essential for achieving social equity [Williamson et al., 2010].

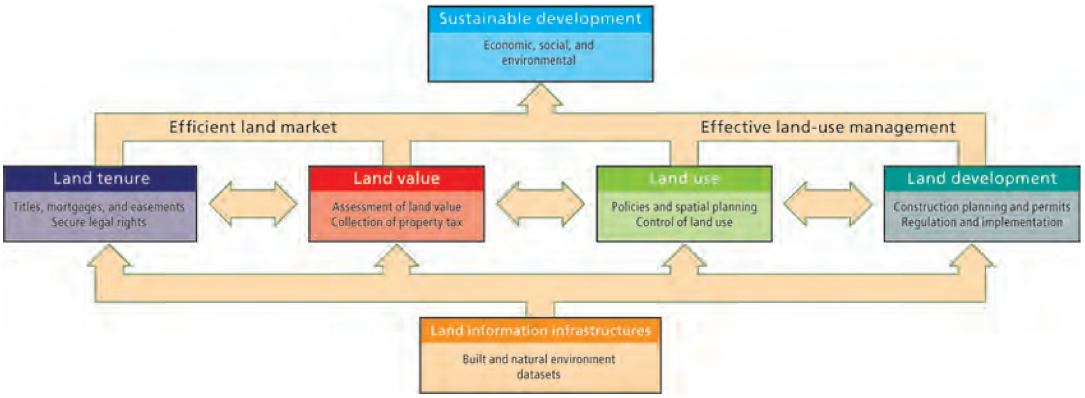


Figure 2.2.: A global land administration perspective promotes sustainable development through efficient land markets and effective land management. [Enemark et al., 2005]

Specifically, several specific SDGs are closely tied to land administration. For SDG 1 (No Poverty) [Lawlor et al., 2019], secure land tenure and equitable land distribution empower individuals by providing them with economic assets, contributing to poverty reduction. In the context of SDG 2 (Zero Hunger) [Mengesha et al., 2022], well-defined land rights enhance agricultural productivity and ensure food security through sustainable farming practices. SDG 5 (Gender Equality) benefits from land administration by promoting equal land ownership for women, which fosters economic empowerment and reduces gender-based inequalities [Agarwal, 2018; Feyertag et al., 2021; Namubiru-Mwaura, 2014]. Furthermore, for SDG 11 (Sustainable Cities and Communities), effective urban land governance supports sustainable urbanization and resilient city planning [Mudau et al., 2020]. In terms of SDG 13 (Climate Action), sustainable land management practices help mitigate the impacts of climate change and promote environmental resilience [Matsumoto et al., 2019]. Lastly, SDG 15 (Life on Land) is advanced by secure land tenure, which is essential for forest conservation, combating desertification, and preserving biodiversity [Mengesha et al., 2022]. These

linkages underscore the fundamental importance of land administration in achieving the economic, social, and environmental dimensions of sustainable development.

Building on these crucial connections between land administration and specific SDGs, global experiences further illustrate how innovative land administration policies and practices have successfully advanced sustainable development in various contexts:

1. Optimizing Land Management through Technology to Support Urban Development and Environmental Protection:
   Some countries have leveraged advanced cadastral systems and SDI to optimize land use, supporting sustainable urban development and environmental protection. For example, the Netherlands has successfully promoted SDG 11 (Sustainable Cities and Communities) and SDG 13 (Climate Action) through its well-developed cadastral system and SDI [Indrajit, 2019]. Similarly, Germany has implemented a Land Degradation Neutrality policy, promoting sustainable land and soil management, which supports SDG 15 (Life on Land) [Wunder et al., 2018].

2. Enhancing Land Tenure Security and Digital Management to Support Poverty Reduction and Economic Development:
   Several developing countries have improved economic development and social equity through digital land management and land tenure registration. For example, India's Digital India Land Records Modernization Programme (DILRMP) has digitized land records, reducing disputes and supporting SDG 1 (No Poverty) and SDG 2 (Zero Hunger) [Choudhury and Behera, 2017]. In Ethiopia, the Rural Land Registration and Certification Project (LRCP) has enhanced land tenure security, promoting SDG 1, SDG 2, and SDG 5 (Gender Equality) [Mengesha et al., 2022]. Rwanda's national land registration project has improved economic conditions in rural areas, advancing SDG 8 (Decent Work and Economic Growth) and SDG 10 (Reduced Inequalities) [Bizoza and Opio-Omoding, 2021; Tan et al., 2021].

3. Addressing Climate Change and Ecosystem Protection through Innovative Land Administration:
   Land administration plays a critical role in addressing climate change and protecting ecosystems. In Brazil, the Amazon Fund has enforced strict land management to reduce illegal logging and deforestation, supporting SDG 13 (Climate Action) and SDG 15 (Life on Land) [Stabile et al., 2020]. Similarly, Greece's Fit-for-Purpose Land Administration has improved land management efficiency, promoting land tenure security and environmental sustainability in urban areas, supporting SDG 1, SDG 2, and SDG 15 [Kalfas et al., 2023].

4. Using Technology and Policy to Secure Rights to Unregistered Land, Promoting Social Equity and Environmental Protection:
   Some countries have used technology and policy to secure the rights of unregistered landholders, promoting social equity and environmental protection. For instance, Kenya has employed remote sensing technology to secure rights to unregistered land, supporting SDG 1 (No Poverty) and SDG 15 (Life on Land) [Koeva et al., 2020]. In sub-Saharan Africa, land management has been key in addressing climate change and ensuring food security, driving progress towards multiple SDGs [Mbow, 2020]. Southeast Asian countries have incorporated land management into their SDG frameworks, particularly in promoting economic growth and social equity [Tirumala and Tiwari, 2022].

These practices demonstrate that land administration plays a critical role in fostering balanced development between economic growth, environmental protection, and social equity.

2.2. Land Administration and ISO 19152 LADM

Land administration is fundamental to securing land tenure, promoting sustainable land use, and supporting socio-economic development. The adoption of standardized frameworks, such as ISO 19152, commonly known as the Land Administration Domain Model (LADM), enhances the efficiency and interoperability of land administration systems by providing a unified structure for managing land-related information across various jurisdictions. This section provides an overview of land administration processes and LADM.

2.2.1. Overview of Land Administration

Land administration refers to the processes involved in determining, recording, and disseminating information about land ownership, value, and use, all to implement land management policies. It focuses on managing land tenure (ownership), land value (taxation and valuation), land use (planning), and land development (infrastructure) [Williamson et al., 2010] which is shown in Figure 2.3. The land administration system(LAS) provides the infrastructure for implementing land policies and strategies. It includes institutional arrangements, legal frameworks, standards, processes, and technologies for managing land tenure, value, use, and development [Steudler et al., 2004].

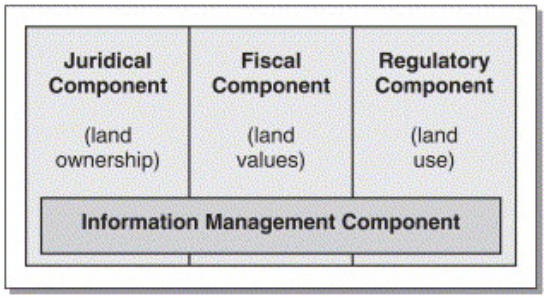


Figure 2.3.: The four basic components of land administration [Steudler et al., 2004]

It is important to distinguish land administration from land management and cadastre, as these terms are often used interchangeably but have differences. Land management encompasses the broader set of policies, strategies, and processes aimed at the sustainable utilization of land resources. It involves decision-making about how land is used, managed, and developed, often at the policy and strategic levels, and focuses on ensuring that land resources meet the current and future needs of society, the economy, and the environment [Dale and McLaughlin, 2000]. The cadastre or cadastral system is an important part of land management and is a specialised land information system. It records detailed information about parcels of land, including their boundaries, ownership and value, and is used to determine who owns what land, where, when and how. Cadastres can be multi-purpose, combining judicial, fiscal and regulatory elements. Cadastral systems provide a spatial basis

for land administration by mapping and registering land units to ensure the legal security of tenure [Hull et al., 2020]. While cadastral systems typically focus on spatial data and property boundaries, land administration is broader in scope, encompassing not only the cadastre but also the legal, institutional and operational processes that govern land tenure and use.

LAS can be broken down into four main functions: land tenure, land value, land use, and land development (Figure 2.2. Land tenure refers to the recording and protection of land rights, ensuring individuals or entities have secure and recognized claims to land. Land value involves the assessment and taxation of land based on its market worth, providing a revenue base for governments and contributing to equitable resource allocation. Land use focuses on planning and regulating how land is utilized, ensuring alignment with broader social, environmental, and economic goals. Land development governs the construction and infrastructure projects on land, ensuring sustainable growth and urbanization [Bennett et al., 2012].

Despite the critical importance of land administration, many countries continue to experience difficulties in implementing LAS. Outdated land records, inconsistent data and limited access to land information are common problems, especially in developing countries. In addition, the land rights of marginalised communities are often not formally recognised, leading to insecurity and conflict [Bennett et al., 2021]. To address these issues, international standards, such as ISO 19152, have been developed to provide a structured framework for land administration to guide the modernisation of land governance systems and to ensure that they meet global standards of efficiency, transparency and equity.

2.2.2. Introduction to ISO 19152 LADM

The Land Administration Domain Model (LADM) is an international standard designed to represent the relationships between people and land by offering a common vocabulary (ontology) and a formal language, namely Unified Modeling Language (UML) [ISO, 2012]. The basic classes of the core LADM are shown in Figure 2.4 LADM plays a key role in improving interoperability between different land management systems and ensuring that the same terminology and data structures are used globally.

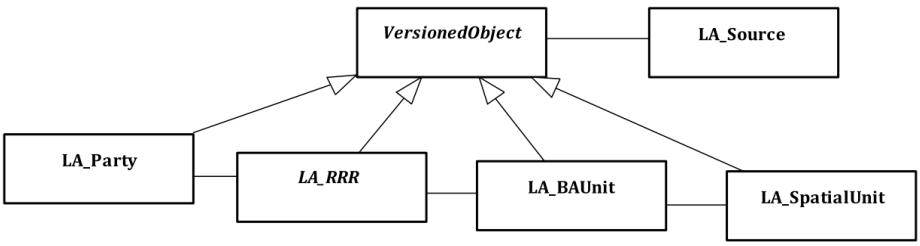


Figure 2.4.: Basic classes of the core LADM [for Standardization , ISO]

The LADM has been widely embraced by international organizations, such as the United Nations [Enemark et al., 2016] and the World Bank, providing a unified framework for diverse stakeholders, including land surveyors [Aditya et al., 2021], land registrars [BECK et al., 2021], and land managers [Lisjak et al., 2021]. To date, approximately ten countries,

such as Scotland, Indonesia, and Colombia, have either implemented or are in the process of incorporating LADM into their land administration systems. Additionally, over fifteen countries have adopted the Social Tenure Domain Model (STDM) [Augustinus, 2010], a related model focused on informal tenure and also a specialisation of the LADM [Lemmen, 2010], shown in Figure 2.5.

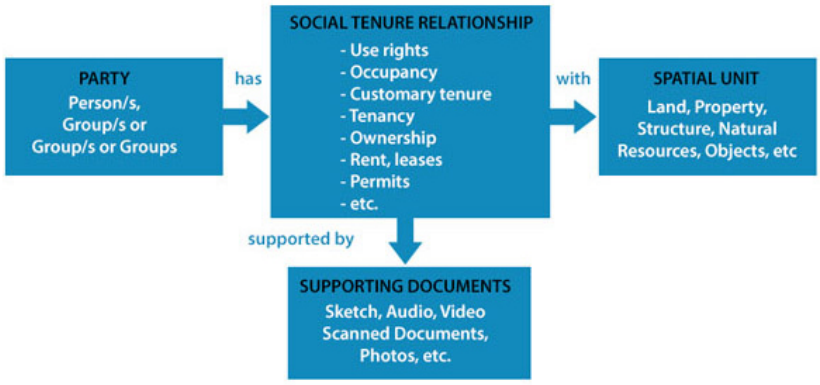


Figure 2.5.: the Social Tenure Domain Model (STDM) [Global Land Tool Network, 2014]

Released in 2012 as ISO 19152, LADM Edition I aimed to standardize land administration systems by providing a common language and framework. It is limited to the land tenure component of the land administration paradigm (see the grey circle in Figure 2.6). Its primary goal is to ensure interoperability between different land administration systems, allowing various stakeholders to manage and exchange land information effectively. LADM Edition II aims to extend the scope of Edition I to include land value, land use and land development (red circle in Figure 2.6 [ISO, 2019]).

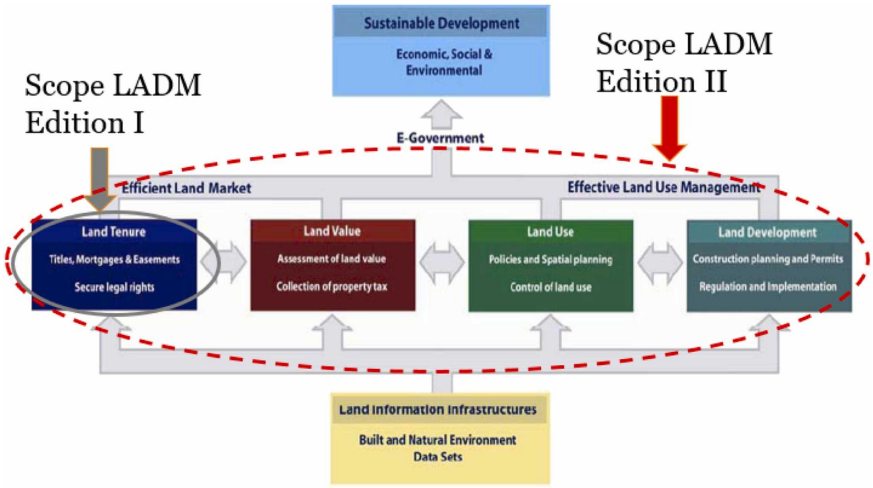


Figure 2.6.: Land administration paradigm and LADM scope ([Kara et al., 2024]; adapted from [Williamson et al., 2006]).

Currently, LADM Edition II is under development, incorporating feedback and technological advancements since the first edition. This revision was prompted by the 2017 UN-GGIM Expert Group Meeting, which highlighted the need for an updated version to better support tenure security and expand land administration coverage. The new edition includes extensions for marine cadastre, 3D/4D cadastres, and refined handling of temporal aspects and RRR (rights, restrictions, responsibilities). It also aims to align with emerging technologies such as blockchain, AI, and SDI. This edition will consist of six parts (Figure 2.7), each addressing specific aspects of land administration. Part 1 offers the overall framework, while Part 2 expands 3D spatial profiles, and Part 3 integrates maritime concepts with land administration. Part 4 covers land valuation, Part 5 focuses on spatial planning, and Part 6 provides guidelines for implementation, emphasizing collaboration with the Open Geospatial Consortium (OGC).The first part of Edition II was recently published [for Standardization , ISO]. The need for improved interoperability with other standards, enriched RRR definitions, enhanced survey models, and more detailed spatial unit representations, including legal spaces in buildings, are key areas addressed in this edition.

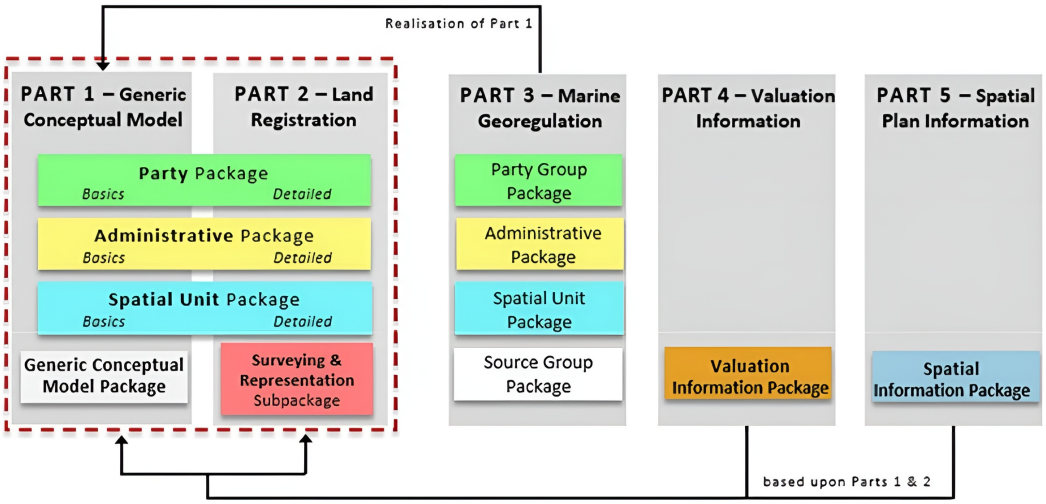


Figure 2.7.: LADM Edition II Parts 1-5 ([Kalogianni et al., 2023]).

2.3. Previous Research Overview

In recent years, there has been increasing recognition of the importance of land administration and its role in supporting the SDGs. As an international standard, LADM primarily deals with managing rights, restrictions, and responsibilities in land administration systems, with the core aim of promoting interoperability between land information systems across different countries and regions.

In terms of standardisation research, Stubkjær et al. [2018] highlighted the importance of LADM's code list management in internationalisation, especially in terms of multilingual support and semantic relations, which are factors that must be taken into account when LADM is applied globally. Similar studies have also shown that the formalisation of code

lists plays a key role in ensuring LADM implementation across jurisdictions and helps to reduce technical barriers due to regional differences [Kara et al., 2022]. Alattas et al. [2018b] developed a database model for indoor navigation by integrating LADM with IndoorGML. Their study addressed key challenges in transforming conceptual models into technical implementations, such as handling primary and foreign keys, constraints, and data types. Their research underscored that while automated tools were used to convert LADM into technical models, manual adjustments were still required to ensure accuracy. In the context of the global implementation of LADM, Kalogianni et al. [2017] proposed an approach based on the INTERLIS language to integrate legal and physical attributes of 3D spaces, addressing semantic interoperability issues when implementing LADM in different countries. They emphasized the use of Semantic Web technologies, such as RDF and OWL, to enhance the definition of code lists and constraints, supporting the extension of LADM.

Although there have been attempts to standardise aspects of LADM, such as code lists and semantic interoperability, there has been little focus on the formalization of land-related SDG indicators using these standards. Existing research has primarily concentrated on technical implementations and standardization within land administration systems, but the application of these standards to consistently define, structure, and calculate SDG indicators, remains unexplored. This gap highlights the need for further research on how international standards like LADM can be leveraged to support the formalization of SDG indicators, ensuring consistency and comparability across regions. Therefore, this thesis aims to address this gap by investigating how ISO 19152 LADM can be applied to the formalization of land-related SDG indicators, making their calculation more systematic.

3. Methodology

3.1. Research Approach Overview

The overall approach of this research follows a structured process that first connects the LADM standard with SDG indicators, and then transitions from a conceptual model to a physical implementation. As shown in Figure3.1, the connection between LADM and the SDG indicators is established using the Four-Step Method, which was developed in previous research. This method will only be briefly introduced in this thesis without detailed analysis. The Four-Step Method helps to construct a conceptual model by systematically mapping key elements of the SDG indicators to the core classes and structures of LADM.

Once the conceptual model is built, the focus shifts to its transformation into a physical data model. This involves implementing the conceptual model using database technologies like PostgreSQL and PostGIS. The physical model incorporates custom data types, standardized code lists, as well as triggers and functions to ensure data integrity and handle complex constraints.

This approach aims to create a seamless transition from conceptual understanding to technical implementation, ensuring the standardization, consistency, and scalability of land-related SDG indicator calculations.

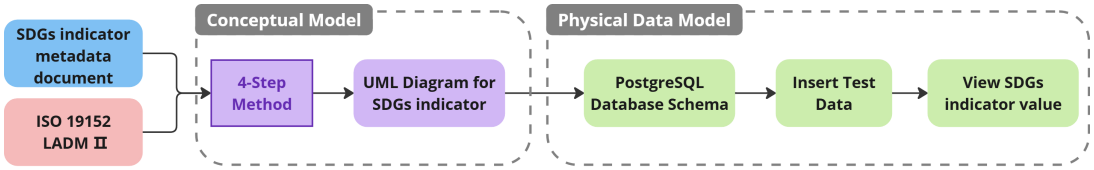


Figure 3.1.: Workflow for Transition from Conceptual Model to Physical Data Model.

3.2. Linking LADM and SDGs Using the Four-Step Method

The Four-Step Method, which was developed as part of my previous research in the Honor Master Program and elaborated in [Chen et al., 2024], provides a structured approach for formalizing SDG indicators within the ISO 19152 LADM framework. This method simplifies the transformation of abstract SDG indicators into a more structured form, enhancing data interoperability and computation for land-related indicators. As shown in Figure 3.2, the method consists of four key steps: keyword Extraction, Matching with LADM concepts, Categorization, and finally Creating UML. Since this method has already been thoroughly analyzed in previous work, only a brief overview will be provided, with further details available in the original paper.

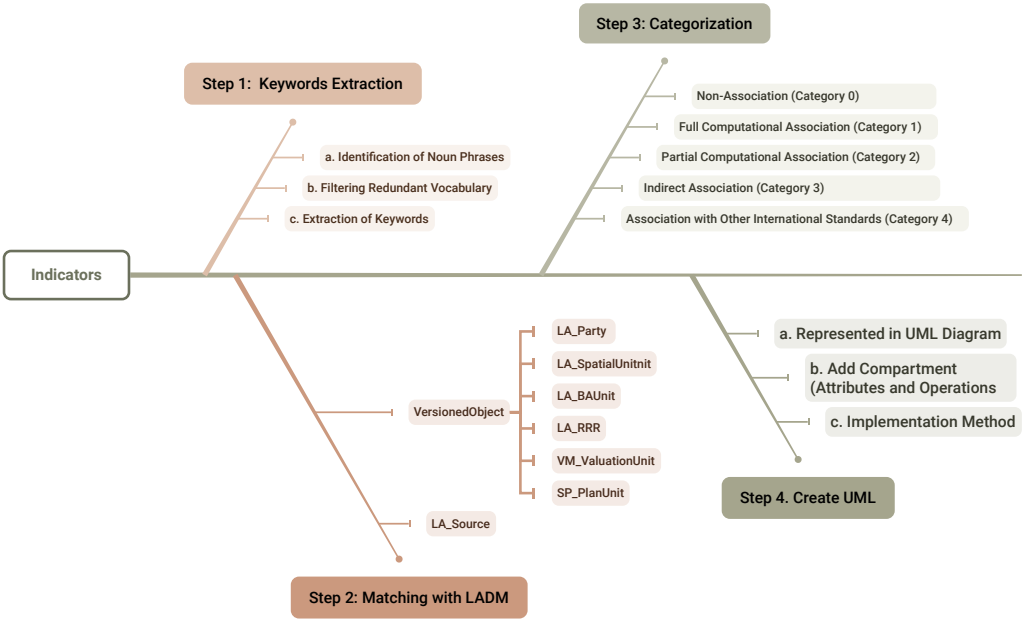


Figure 3.2.: The Four-Step Method. [Chen et al., 2024]

3.2.1. Step 1: Keyword Extraction

The first step in the Four-Step Method involves filtering SDG indicators based on core LADM terminology. The key LADM terms include Land, Party, RRRs (Rights, Responsibilities, Restrictions), Spatial Units, Marine, Valuation, and Spatial Plan. These terms are defined in detail according to their roles within the LADM framework. These keywords provide the foundation for aligning the indicators with the LADM components.

3.2.2. Step 2: Matching with LADM concepts

After extracting the relevant keywords, the next step involves mapping these keywords to corresponding LADM classes. The process begins with a detailed analysis of the SDG indicator metadata documents, focusing on key sections such as ”Related Indicators,” ”Definition and Concepts,” ”Data Sources,” and ”Method of Computation.” These sections help clarify the relationships between indicators, define key concepts, evaluate relevant data sources, and determine computation methods. Following this, a rigorous alignment process is conducted to map the SDG indicators to the core LADM classes, including LA_Party, LA_SpatialUnit, LA_BAUnit, LA_RRR, VM_ValuationUnit, and SP_PlanUnit, all of which are derived from VersionedObject and are associated with LA_Source. This mapping process formalizes the relationships between the SDG indicators and the core components of the LADM model. For any data that LADM cannot provide, it is noted that external datasets may be required. This thorough analysis and matching process ensure the technical accuracy and conceptual reliability of the selected indicators.

3.2.3. Step 3: Categorization

After matching the keywords with LADM classes, the method proceeds to categorize the level of association between the SDG indicators and the LADM components. This step ensures that the relationships between SDG indicators and the land administration domain are clearly defined. Indicators are categorized based on whether they can be fully or partially computed using the existing LADM classes, or if they require additional components, such as external classes or custom attributes, to accurately represent the SDG requirements.

3.2.4. Step 4: Create UML

In the final step, the formalized SDG indicator is transformed into a UML diagram, providing a visual representation of how the indicator can be computed within the LADM framework. This involves creating a complete UML model that includes the core LADM components and any necessary external classes. The model serves as the foundation for implementing SDG indicator calculations in a physical data model, ensuring that the formalized indicators can be accurately computed and monitored in practice.

Specifically, the UML diagram supports the calculation of the indicator by representing three types of information: (a) information that can be directly represented by the LADM package, (b) information that can be connected to other databases via external classes, and (c) information that will be output through interface classes. For relevant classes, a dedicated compartment is added to perform the calculation operations for the indicator, which includes parameters such as year and region, allowing for computation across different temporal and spatial scopes. Additionally, each operation is clearly defined with specific implementation methods in the UML diagram, with the implementation steps described using programming languages such as Python, Java, or pseudocode. The final results are aggregated and presented through interface classes, similar to other classes in LADM used for summarizing information, such as LA_SpatialUnitOverview or LA_PartyPortfolio.

This systematic approach ensures a structured and comprehensive documentation of SDG indicator development and computation, while promoting transparency and clarity in the calculation processes associated with the LADM framework.

In addition to these core steps, the UML diagram created in the final step helps bridge the gap between conceptual modeling and physical data implementation. It formalizes the computational rules and interfaces needed to calculate SDG indicators, while also providing flexibility for incorporating external data sources and additional parameters. By following this method, land-related SDG indicators can be systematically integrated into land administration systems, promoting accurate and efficient data analysis.

For a more comprehensive explanation of the Four-Step Method and its specific application in SDG indicator calculation, please refer to the author's previous publication [Chen et al., 2024], which discusses each step in detail and provides several examples using different parts of LADM Edition II.

3.3. Transition from Conceptual to Physical Model

In this research, the transition from a conceptual model to a physical model is a crucial step to ensure that the system can support the calculation and analysis of land-related data. The physical model is built based on the LADM standard framework and uses database technologies to manage and analyze land-related data. The following sections describe the strategies and standards adopted in the process of transitioning from the conceptual model to the physical model.

3.3.1. Physical Model Requirements

During the transition from the conceptual model to the physical model, the physical model must meet several key requirements to ensure standardization, consistency, and flexibility in data processing. These requirements include:

- **Data Consistency**: The physical model must ensure the consistency of land-related data during operations such as insertion, updating, and deletion. This requires the physical model to include comprehensive constraint mechanisms and validation rules to prevent operations that do not comply with the standards from affecting data accuracy.
- **Standardization**: The model design must conform to the ISO 19152 LADM standard to ensure global compatibility and interoperability in land data management. By using standardized data types and class structures, the physical model can seamlessly integrate with other LADM-based systems.
- **Scalability**: The design of the physical model should support flexible expansion to accommodate additional land tenure types, data sources, and future computational needs. The architecture of the physical model must be capable of handling future data growth without requiring significant modifications to the fundamental structure.
- **Operability**: The abstract concepts defined in the conceptual model must be effectively implemented in the physical model, ensuring that these concepts are operational and can support efficient data storage, retrieval, and computation in practice.

These requirements ensure that the transition from conceptual to physical models focuses not only on the accuracy of the model but also on the usability and long-term maintenance of the system.

3.3.2. Model Transformation Strategy

The design of the physical model involves extracting key elements from the conceptual model and mapping them into specific data structures and database operations. The following strategies were adopted to achieve this transition:

- **Class-to-Table Mapping**: In the conceptual model, LADM classes (such as LA_Party, LA_SpatialUnit, etc.) are defined as abstract class structures. The physical model translates these classes into concrete database tables, ensuring that data storage and operations comply with the standard definitions of LADM. Each class is mapped to

one or more tables, while maintaining the integrity of relationships between classes, for example, by using foreign key constraints to preserve associations between entities.

- **Attribute-to-Column Mapping**: Each attribute of an LADM class is mapped to fields in the database table in the physical model. Standard data types are used to define these fields in the physical model to ensure compatibility with the LADM model. Additionally, certain specialized attributes may require custom data types to meet specific needs.

- **Normalization**: The design of the physical model follows the principles of database normalization to reduce data redundancy and ensure efficient data storage. By decomposing the classes and relationships in the conceptual model into multiple related tables, the physical model ensures data consistency and integrity.

- **Hierarchical Mapping**: Some classes in the LADM model have hierarchical relationships (e.g., versioned objects). In the physical model, these relationships are implemented through hierarchical table structures. For example, relationships between derived classes and base classes can be maintained through inheritance mechanisms or composite primary keys, ensuring consistency and data traceability.

Through these strategies, the physical model successfully translates the theoretical structure of the conceptual model into an operational database model, while ensuring accuracy and standardization in the process.

3.3.3. Ensuring Data Consistency and Integrity

Data consistency and integrity are core requirements in the design of the physical model. To ensure these requirements are met, several methods were adopted during the model transformation process to safeguard data accuracy:

- Data **Constraints**: By designing strict constraints in the database (such as primary keys, foreign keys, unique constraints, and check constraints), the physical model can ensure the validity of each data entry. For example, each land unit (LA_SpatialUnit) must have a unique identifier, and its related tenure information must conform to pre-defined logical relationships.

- **Hierarchical Data Integrity**: Versioned objects in LADM (such as VersionedObject) require proper management of both historical and current data. Through the design of hierarchical table structures in the physical model, the traceability of historical data and the consistency of current data are ensured.

- **Transaction Control**: The physical model defines a series of transaction operations to ensure data atomicity and consistency. This means that each operation (e.g., inserting, updating, or deleting data) must be completed in accordance with integrity constraints, or the system will roll back the operation to prevent inconsistent data from being saved.

These data consistency and integrity mechanisms ensure that the physical model can reliably manage land-related data operations, preventing data loss or logical errors.

3.3.4. Scalability and Flexibility

Scalability and flexibility were important considerations during the design of the physical model. Several measures were taken to ensure that the model can accommodate future changes in demand:

- **Modular Design**: The design of the physical model is highly modular. Each LADM class or component is treated as an independent module that can be expanded or modified as needed. This design ensures that the system can flexibly add new features or class structures based on the land administration needs of different countries or regions without requiring major changes to the existing model.
- **Support for External Data Integration**: The physical model was designed to accommodate the integration of external data sources. For example, some land tenure data may come from external systems, and the model provides interface classes and foreign key references to achieve seamless integration with other databases or external systems. This flexibility allows the physical model to handle different types of data inputs and outputs.
- **Scalability Consideration**: To ensure that the model can handle future data growth, the physical model adopts a scalable architecture design. Through partitioning tables and optimizing indexing, the physical model can efficiently manage large datasets without compromising system performance.

This theoretical framework serves not only as a design principle for the transition from a conceptual model to a physical model but also as a guiding methodology for standardizing land administration systems. By adhering to this structured approach, the design ensures that the physical model remains consistent with international standards while providing the necessary flexibility and scalability for future expansions. This framework can be applied as a foundational guideline for the development of land information systems, supporting robust and reliable data management and computation across various land-related domains.

3.4. Conceptual Model for SDG Indicator 1.4.2

In this thesis, SDG 1.4.2 is used as an example to demonstrate the development of both the conceptual and physical models. The conceptual model for SDG 1.4.2 had already been developed in previous research, where it was systematically constructed following the Four-Step Method. However, the primary focus of this thesis is on the implementation of the physical model. Nevertheless, to provide a comprehensive overview and to illustrate the process in a detailed manner, the Four-Step Method is applied again here to develop the conceptual model for SDG 1.4.2, ensuring that each step is clearly demonstrated in relation to the chosen indicator.

The decision to focus on SDG 1.4.2 stems from its critical role in assessing land tenure security. This indicator not only tracks the proportion of the population with documented ownership or legal land use rights but also captures individuals' perceptions of tenure security, a key factor in sustainable land governance. By addressing both legal and perceived land tenure, SDG 1.4.2 aligns directly with the core objectives of land administration. Furthermore, its structure is highly compatible with the ISO 19152 LADM standard, as it integrates

seamlessly with key LADM components, making it an ideal example for demonstrating the systematic calculation and analysis of land-related indicators.

3.4.1. Introduction to SDG 1.4.2

SDG Indicator 1.4.2: "Proportion of total adult population with secure tenure rights to land, (a) with legally recognized documentation, and (b) who perceive their rights to land as secure, by sex and type of tenure."

SDG 1.4.2 aims to measure the proportion of the population with secure tenure rights to land, including those who hold documented ownership or legal land use rights, as well as the perception of such rights. This indicator plays a crucial role in monitoring progress towards achieving land tenure security. The key components of SDG 1.4.2 include:

- **Documented Land Rights**: Refers to the proportion of the population holding formal documentation that recognizes their rights to land, such as titles, deeds, or other legal evidence.
- **Perceived Security of Tenure**: Captures individuals' perceptions of the security of their land rights, which is essential for assessing tenure security, even in the absence of formal documentation.
- **Disaggregation**: Data for SDG 1.4.2 is often disaggregated by variables such as gender, age, geographic location, and land type (e.g., agricultural, residential, or communal).

3.4.2. Applying the Four-Step Method to SDG 1.4.2

To construct the conceptual model for SDG 1.4.2, the Four-Step Method is applied, specifically aligning with Part 2 of the second edition of the LADM, which focuses on land registration. The following sections outline each step of this process.

Step 1: Keyword Extraction

The process of "Identification of Noun Phrases–Filtering Redundant Vocabulary–Extraction of Keywords" was used, as shown in Figure 3.3. The keywords include "legally recognized documentation", "adult population", "sex", "secure tenure rights", "rights to land" and "type of tenure", and their corresponding LADM core terms are "Source", "Party" and "Rights".

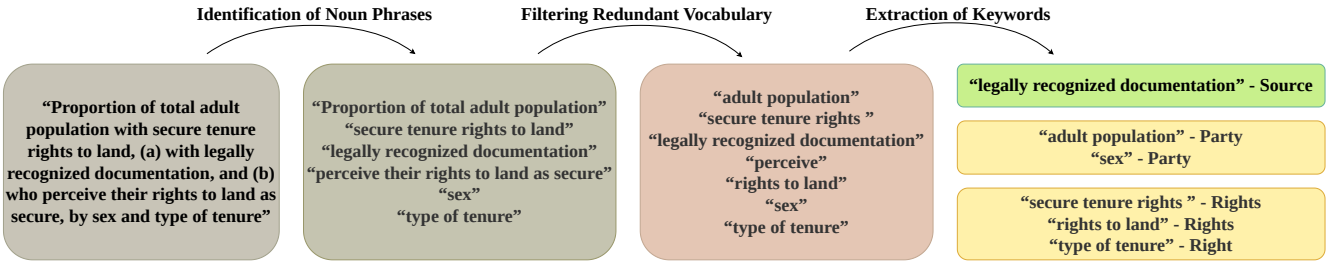


Figure 3.3.: Keyword extraction for SDG 1.4.2.

Step 2: Matching with LADM concepts

The SDG Indicator 1.4.2 is divided into two parts according to its metadata: (A) the proportion of adults with legally recognized documentation over land, and (B) the proportion of adults who perceive their land rights as secure. To meet these requirements, three types of data are needed: (1) the number of adults with legally recognized documentation (derived from LADM-compliant land administration systems), (2) the number of adults who perceive their rights as secure (collected via surveys or external data), and (3) the total adult population (from census data).

$$\text{Part (A)}: \frac{\textit{People (adult) with legally recognized documentation over land}}{\textit{Total adult population}} \times 100$$

$$\text{Part (B)}: \frac{\textit{People (adult) who perceive their land rights as secure}}{\textit{Total adult population}} \times 100$$

The following LADM components were matched with SDG 1.4.2 concepts:

- **LA_RRR** is related to "secure tenure rights" and "type of tenure," as these reflect the nature of land rights.
- **LA_Source** corresponds to "legally recognized documentation," involving the legal registration of rights.
- **LA_Party** includes the "sex" attribute, as it relates to disaggregation by gender.

Step 3: Categorization

SDG 1.4.2 is classified under the "Partial Computational Association (Category 2)".

Step 4: Create UML

The computation of SDG Indicator 1.4.2 is modeled through a UML diagram based on the steps outlined in the indicator development process. The main elements are as follows:

1. **Represented in the UML Diagram**: The UML diagram includes core LADM classes such as LA_Party, LA_RRR, LA_BAUnit, and LA_Source. Two external classes, ExtSecureLandRightAdult and ExtParty, are introduced to represent Part B of the indicator (perceived tenure security) and to capture additional party-related data.
2. **Add Compartment (Attributes and Operations)**: Each class in the UML diagram is assigned compartments containing specific attributes and operations necessary for the computation of the indicator, as outlined in Table 3.1. In addition, new classes specific to SDG 1.4.2 have been introduced to accommodate the requirements of the indicator:
    - **ExtParty**: This external class includes attributes such as *birthday*, which is used to determine adulthood, and the operation *countAdult* to calculate the total adult population.
    - **ExtPartyPerceiveSecureLandRights**: This class includes attributes like *selfPerception*, which captures the perceived security of land rights (with values of 1 indicating secure rights and 0 indicating insecurity), sourced from household surveys.

- **Interface Class SDG_1.4.2**: This interface aggregates information from **LA_Party**, **LA_RRR**, and **LA_Source**, and includes operations such as *computeProportionWithLegalDocumentation* and *computeProportionPerceivingSecurity*. These operations compute the proportions of adults with legally recognized land rights and those who perceive their rights as secure, respectively.

3. **Computation Methods**: The methods designed to compute the indicator include computeProportionWithLegalDocumentation, computeProportionPerceivingSecurity, generateReport, and countAdults. These methods ensure that the theoretical framework is implemented into practical algorithms to calculate the required proportions and generate reports. The final UML diagram for SDG 1.4.2 is shown in Figure 3.4.

Existing LADM Class	Attributes Used in the Case	Notes
LA_Party	+gender:LA_HumanSexesType[0..1]	Highlighted to facilitate gender-based classification and calculation.
LA_Right	LA_RightType	Delineating various land tenure types, echoing the ”type of tenure” parameter in the indicator. The specific right types are detailed in the ”Code List.”
LA_AdministrativeSource	+type:LA_AdministrativeSourceType	Signifying ”Legally recognized documentation.” Its code list meticulously enumerates the possible values, such as agriLease, deed, and title.
LA_BAUnit		While not the focal point, it is outlined to underscore the indicator’s emphasis on rights over land.

Table 3.1.: Attributes added to SDG 1.4.2 UML from an existing LADM class.

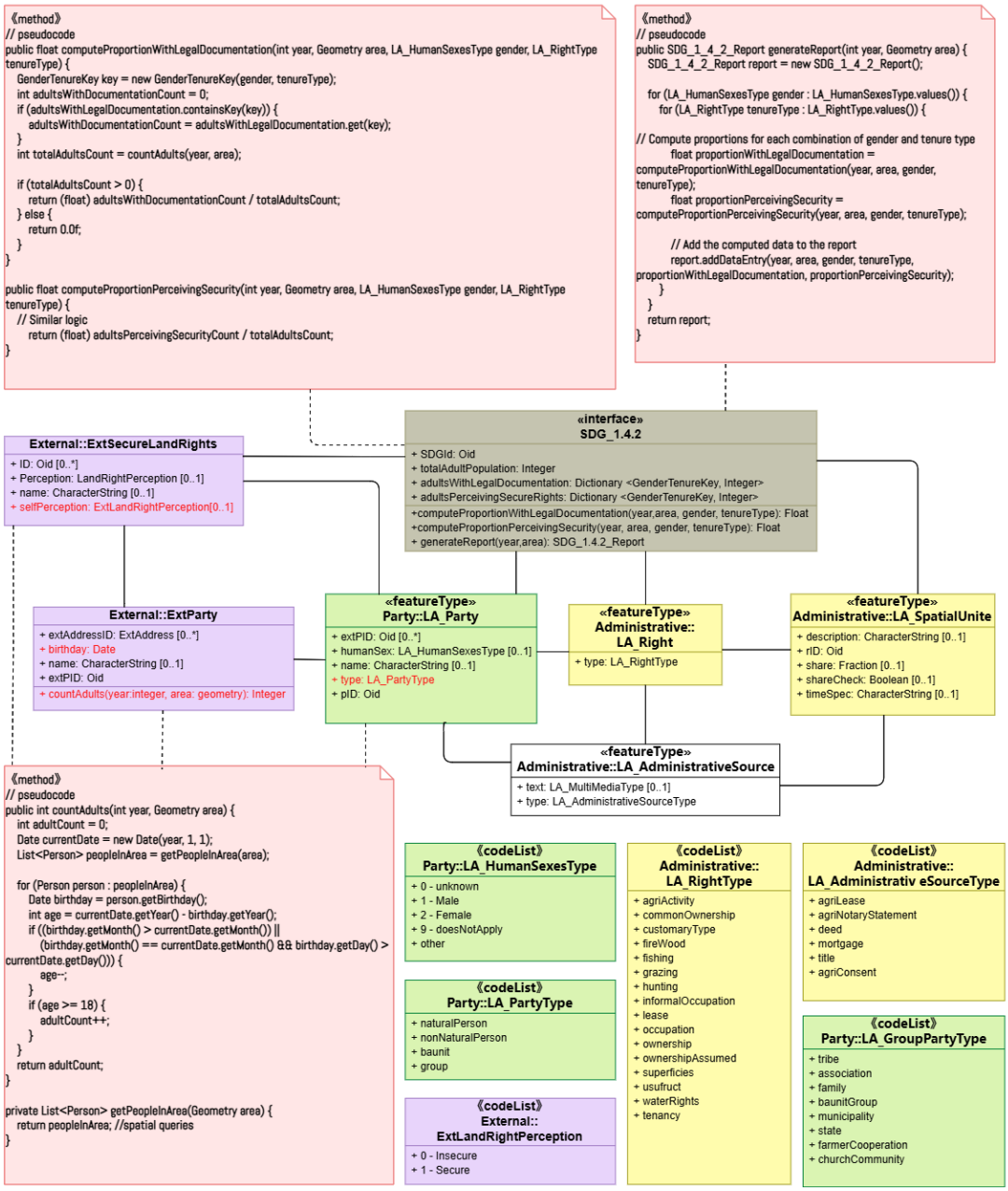


Figure 3.4.: UML Class Diagram for SDG Indicator 1.4.2 Based on the LADM Model).

4. Implementation

The previous sections outlined the overall research approach, specifically detailing the Four-Step Method in Linking LADM and SDGs Using the Four-Step Method and the transition process in Transition from Conceptual to Physical Model. These sections provided a comprehensive overview of the methodology and summarized the outcomes of previous research, where the Four-Step Method was applied to specific SDG indicators. In this case, SDG 1.4.2 was chosen as the focal indicator, and the final UML diagram was produced as a result.

This chapter focuses on how to apply the physical model methodology described in the previous chapter to develop a database specifically for the calculation and management of SDG 1.4.2. The following sections will describe the implementation process, including the creation of custom data types, the use of functions and triggers for data integrity, and the generation of reports using the database.

4.1. Overview of Tools Used

To implement the physical data model, a combination of tools was selected to support the specific needs of this project. The key tools used include PostgreSQL, PostGIS, and QGIS, each chosen for their ability to facilitate different stages of the modeling process, from database creation to spatial data visualization for reporting purposes.

4.1.1. PostgreSQL and PostGIS

For the relational Database Management System (DBMS), PostgreSQL was selected as the platform for implementing the physical model. The majority of cadastral and land administration data are highly structured, making relational data modeling and a relational DBMS the most suitable choice. Among the widely used options, PostgreSQL/PostGIS and Oracle Spatial are the most prominent. PostgreSQL was chosen primarily due to its open-source nature and its support for spatial applications via the PostGIS extension. [Shahidinejad et al., 2024] PostGIS provides robust support for geographic objects, allowing spatial data to be stored and queried directly within the PostgreSQL environment. This combination seamlessly manages both structured data, such as land ownership information, and unstructured spatial data, such as parcel boundaries, which are critical in land administration. Additionally, PostgreSQL's open-source nature offers cost-effectiveness and flexibility, making it an ideal choice for projects requiring ongoing development and adaptation.

4.1.2. QGIS for Spatial Visualization

QGIS was selected for its superior spatial data visualization capabilities, which are essential for reporting purposes. Since the SDG indicators computed in this study involve spatial data (e.g., parcels and ownership), the final output includes spatial information related to land parcels and administrative boundaries, making QGIS the ideal tool for generating maps and visual reports. Its compatibility with PostgreSQL/PostGIS allows for seamless integration, enabling the visualization of spatial queries and results. This is particularly important for ensuring that the final reports, which involve spatial analysis, are not only accurate but also visually interpretable.

4.1.3. Direct SQL Code Generation from UML Diagram

A critical decision in the database design process was the approach used to generate the database schema from UML diagrams. While tools such as Database Builder in Enterprise Architect (EA) offer built-in functionality to generate Data Definition Language (DDL) scripts directly from UML diagrams [Alattas et al., 2018c,a; Chehrehbargh et al., 2024], this option was not selected. Instead, SQL code was manually generated based on the UML diagrams. The primary reason for this choice was the scope and specificity of the project. At this stage, each SDG indicator has its own dedicated UML diagram, which only utilizes a subset of the full LADM model. The UML diagrams also include external classes that are not fully covered by the LADM, meaning that DDL scripts generated from EA would still require substantial manual adjustments to accommodate these external classes. Given these factors, generating SQL code directly from the UML diagrams allowed for greater flexibility and control over the creation of the database schema. This method also facilitated the manual creation of external classes not included in the LADM, ensuring that all necessary entities and relationships were accurately represented in the final database schema.

While the overall implementation process applies broadly to any land administration-related SDG indicator, this study uses SDG Indicator 1.4.2 as a case example to illustrate the detailed workflow and to validate the feasibility of the previously established conceptual model. The subsequent sections will focus specifically on the implementation of SDG 1.4.2, demonstrating how the conceptual model is transformed into a physical database schema, and how the calculations are conducted. All remaining content in this paper will revolve around SDG 1.4.2, detailing the steps involved in transforming the model into practice.

4.2. Custom Data Types and Constraints Implementation

4.2.1. Custom Data Types

This section focuses on two specialized data types from the LADM model: `Oid` and `Fraction`. These data types are crucial for representing unique spatial objects and allocating shares of land tenure rights. Since there are no direct equivalents for these types in most database management systems, they must be manually created and implemented. By utilizing custom data types and additional constraints, these structures are incorporated into the database schema to ensure accuracy and integrity of land administration data.

Oid

In LADM, the `Oid` (Object Identifier) is a generalized data type used to ensure the uniqueness of spatial objects, particularly in Part 1 of the LADM Generic Conceptual Model. This data type consists of two core attributes: `localId` (local identifier) and `namespace`. The `localId` is assigned by the data provider and must be unique within a specific namespace, while the `namespace` attribute identifies the source of the spatial object. To maintain generality, both `Oid` and `Fraction` are represented as `CharacterString` data types, allowing flexibility across different application contexts.

When implementing `Oid` in a database, this generic data type can be modeled through a custom data type. However, when creating custom data types, databases generally do not allow direct constraints such as `NOT NULL` or `UNIQUE` within the data type definition. Therefore, these constraints must be implemented at the table level during the creation of the entity tables. Custom data types only define attributes and their data types, while constraints must be added through table-level constraints to ensure non-nullability and uniqueness.

Consider further that in PostgreSQL, it is not permissible to directly use a custom `Oid` type as a primary key because PostgreSQL requires primary keys to be a simple data type that is natively supported by the database. To address this, the `localId` and `namespace` attributes are split into separate fields and combined to form a composite primary key. This approach ensures uniqueness while adhering to PostgreSQL's primary key requirements.

For example, the Oid data type can be defined using the following SQL code, containing the `localId` and `namespace` fields:

```
CREATE TYPE Oid AS (
    localId VARCHAR,
    namespace VARCHAR
);
```

While the database does not permit constraints within the custom data type, table-level constraints such as `NOT NULL` and `PRIMARY KEY` can be added during table creation to enforce the non-nullability and uniqueness of the `localId` and `namespace` attributes. Below is an example of how the `Oid` is implemented in the *LA_Party* table:

```
CREATE TABLE LA_Party (
    Pid Oid NOT NULL,  -- Uses composite type Oid
    Pid_localId VARCHAR NOT NULL,  -- Stores localId separately
    Pid_namespace VARCHAR NOT NULL,  -- Stores namespace
       separately
    PRIMARY KEY (Pid_localId, Pid_namespace),  -- Create
       composite primary key
    CHECK (Pid_localId IS NOT NULL AND Pid_namespace IS NOT NULL
       )
);
```

In the above code, the `Pid` field represents an instance of the `Oid` composite type, while the `localId` and `namespace` attributes are stored separately for subsequent addition of constraints. The `PRIMARY KEY` constraint ensures the uniqueness of the combination of

Pid_localId and Pid_namespace, meeting the LADM requirement for Oid uniqueness. Additionally, the CHECK constraint ensures that both attributes are non-null, which further fulfils its cardinality requirement.

However, in practice, it may not be straightforward to use Oid as a custom composite data type. In many cases, the localId and namespace attributes are concatenated into a single string to serve as the object identifier. This approach simplifies database implementation and enhances data processing efficiency. In this study, the second approach was used during data validation.

Fraction

In LADM, Fraction is another general data type, which is specifically designed to support the expression of shares. The Fraction type consists of two core attributes: numerator and denominator. The denominator must be a positive integer greater than zero, while the numerator must be a non-negative integer and cannot exceed the value of the denominator. This representation ensures precise allocation of shares, especially when multiple parties hold equal or unequal shares.

In database implementation, Fraction can be modeled as a custom data type. However, since databases cannot directly enforce constraints on the internal attributes of custom types, trigger functions are used to ensure the validity of the fractions. Specifically, the trigger function checks whether the numerator and denominator of the fraction meet the basic constraints during record insertion or update, ensuring that the total share amounts to 1.

The Fraction data type is defined with numerator and denominator attributes:

```
CREATE TYPE Fraction AS (
    numerator INTEGER,
    denominator INTEGER
);
```

To ensure the validity of the Fraction, a trigger function is implemented to verify that the numerator and denominator adhere to the required constraints. The trigger function checks that the denominator is greater than zero, the numerator is non-negative, and the numerator does not exceed the denominator. This ensures that invalid data does not enter the system. The details are illustrated in section 4.3.2.

Additionally, to ensure that in cases of joint ownership, the total ownership share sums to 1 across all parties, another trigger function is implemented. This function dynamically generates an SQL query to calculate the total share sum by normalizing the fractions and ensuring that the total sum equals 1. This function is applicable to tables like *LA_Right* and *LA_PartyMember*, and ensures the accuracy of shares. The core logic for this check is as follows, more details in sec 4.3.3:

```
CREATE OR REPLACE FUNCTION check_share_sum()
RETURNS TRIGGER AS $$
BEGIN
    -- Calculate the total ownership share sum
    -- If the total sum is not equal to 1, raise an exception
    IF total_numerator <> total_denominator THEN
        RAISE EXCEPTION 'Total share is not equal to 1';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

After the function has been created, triggers need to be set up for the relevant tables (e.g. *LA_Right* and *LA_PartyMember*) to automatically perform this check on inserts or updates. Creating triggers is done in the same way as any other function to ensure that the function gets called when it needs to, which will be discussed in more detail later in the Constraints section. Through the design and implementation of these trigger functions, the database automatically ensures that the Fraction type adheres to LADM constraints and that the total ownership share across parties is accurate. This approach ensures data consistency and maintains the integrity of ownership share distributions. While the code provided illustrates the logic, it is not the final implementation. Full details of the complete code implementation can be found in the appendix.

In addition to the two custom data types described above, most of the data types required by LADM are already supported natively in the database system and can be directly used. The table 4.1 summarizes how LADM attributes map to their corresponding database data types.

This table outlines the LADM attributes and their corresponding data types within the database. Attributes such as name for *LA_Party* or *ExtParty* and timeSpec for *LA_Right* are easily represented as `TEXT` fields in the database, while temporal attributes such as `beginLifespanVersion` and `endRealWorldLifespanVersion` are implemented using `TIMESTAMPTZ` (timestamp with time zone) to capture the required temporal precision as specified in ISO 19108. The `geom` attribute, which represents spatial information for *LA_SpatialUnit*, is handled by the `GEOMETRY` type provided by PostGIS, allowing for efficient storage and querying of spatial data. Additionally, attributes related to code lists, such as humansex or tenure types associated with *LA_Party* and *LA_Right*, are represented as `INTEGER` fields in the database, with associated code list values stored separately in related tables.

Attribute	Class	LADM Data Format	Database Data Format
name	LA_Party, ExtParty, ExtSecureLandRightsQuestionnaire	CharacterString	TEXT
Four Temporal Attributes (beginLifespanVersion, endLifespanVersion, beginRealWorldLifespanVersion, endRealWorldLifespanVersion)	VersionedObject	DateTime (ISO 19108)	TIMESTAMPTZ
birthday	ExtParty		DATE
shareCheck	LA_Right	Boolean	BOOLEAN
timeSpec	LA_Right	CharacterString	TEXT
geom	LA_SpatialUnit	Geometry	GEOMETRY
xx_type / humansex / ExtSecureLandRightsQuestionnaire	LA_Party, LA_PartyGroup, LA_Right, LA_AdministrativeSource	codelistname	INTEGER

Table 4.1.: Mapping of LADM Attributes to Database Data Types.

4.2.2. Standardized and Extensible Code Lists

Code lists play a crucial role in ensuring both standardization and flexibility within the LADM-based data model. A code list provides a controlled vocabulary for specific attributes, ensuring that the values used across different systems are consistent and uniform. This not only enhances semantic interoperability between land administration systems but also improves the ease of querying and data manipulation within the database.

In the context of a database, code lists are implemented as separate tables, where each entry represents a possible value for a particular attribute. These code list values are referenced in other tables to enforce the range of permissible values for certain attributes. For example, the *LA_PartyType* attribute, which describes whether a party is a natural person, non-natural person, or a group, uses a code list to ensure that only predefined values are allowed. This guarantees consistency and avoids data entry errors.

In addition to standardization, code lists also improve operational efficiency within the database. Since the descriptions of these values are typically longer strings, using integer identifiers instead simplifies storage and allows for more efficient querying. This separation of codes and descriptions also simplifies updates to the code list, as new values can be added or old values modified without impacting the data structure.

The implementation of code lists in PostgreSQL can follow two general approaches, depending on whether the values are fixed or flexible:

1. **Fixed Code Lists**: For attributes that use predefined, unchanging values, such as gender or land rights perception, a fixed code list is employed. These values are assigned unique identifiers, which rarely need modification or expansion. For instance, the *LA_HumanSexesType* code list defines the values for gender as `Male`, `Female`, `Unknown`, etc., with corresponding integer identifiers:

```
CREATE TABLE LA_HumanSexesType (
hst_id INTEGER UNIQUE NOT NULL,
hst_description TEXT NOT NULL
);
INSERT INTO LA_HumanSexesType (hst_id, hst_description )
    VALUES
(0, 'unknown'),
(1, 'Male'),
(2, 'Female'),
(9, 'doesNotApply'),
(99, 'other');
```

2. **Flexible Code Lists**: For attributes that require more flexibility or may need to accommodate new values over time, such as the types of parties (*LA_PartyType*) or administrative sources, a flexible code list is employed. These tables use an auto-incrementing serial key, allowing new entries to be added dynamically. For instance, the *LA_PartyType* code list can be defined as follows:

```
CREATE TABLE LA_PartyType (
pt_id SERIAL PRIMARY KEY,
pt_description TEXT NOT NULL UNIQUE
);
INSERT INTO LA_PartyType (pt_description ) VALUES
('naturalPerson'),
('nonNaturalPerson'),
('baunit'),
('group');
```

   In this case, the `pt_id` is automatically generated and serves as the primary key, while the `pt_description` holds the textual representation of the party type. This method allows for future extensibility, enabling jurisdictions to add new types of parties or administrative entities as needed.

Code Lists as Attribute Constraints

In the LADM model, certain attributes are directly linked to their corresponding code lists, with the attribute's value being constrained to the possible values defined within the code list. For example, the type attribute of the *LA_Party* class refers to the *LA_PartyType* code list, meaning that its value can only be selected from the predefined types defined in the *LA_PartyType* table. In the database schema, this relationship is enforced by creating a foreign key constraint that links the attribute to its code list:

```
    CREATE TABLE LA_Party (
    -- Other sql code
    p_type INTEGER NOT NULL REFERENCES LA_PartyType(pt_id)
);
```

Here, the p_type attribute in the *LA_Party* table represents the party's type, and its value is restricted to the set of valid pt_id values in the *LA_PartyType* table. Additionally, because the cardinality of this attribute in LADM is 1 (i.e., a party must have exactly one type), the NOT NULL constraint is applied to ensure that every party has a valid type assigned.

This pattern of using integer identifiers from code lists as foreign keys is applied consistently across the LADM database schema. It ensures that attributes such as gender, tenure type, and party type are restricted to valid, predefined values, which maintains data integrity and enforces uniformity across the system.

Flexibility and Extensibility

Although code lists provide a standardized set of values, they are designed to be both flexible and extensible. Jurisdictions can add local values to the code lists if needed, allowing the LADM implementation to adapt to specific local or national requirements. This flexibility is essential for accommodating the diverse legal, administrative, and social systems found across different regions or countries.

For instance, in the case of *LA_GroupPartyType*, the initial code list might include basic values such as tribe, association, family, and baunitGroup. However, in some jurisdictions, there may be additional categories of party types that need to be included, such as farmerCooperation or churchCommunity. Rather than modifying the structure of the database schema, these new categories can be added directly to the code list, as shown below:

```
-- Original code list
CREATE TABLE LA_GroupPartyType(
gpt_id SERIAL PRIMARY KEY,
gpt_description TEXT NOT NULL UNIQUE
);

-- Insert original values
INSERT INTO LA_GroupPartyType (gpt_description ) VALUES
('tribe'),
('association'),
('family'),
('baunitGroup');

-- Adding new values
INSERT INTO LA_GroupPartyType (gpt_description ) VALUES
('farmerCooperation'),
('churchCommunity');
```

In this example, the database schema remains unchanged, and the system can continue functioning without any interruptions. The new values can now be referenced in the relevant *LA_GroupPartyType* entries, ensuring that the system accurately reflects the legal and administrative realities of the specific region. This approach maintains the integrity and consistency of the LADM implementation while allowing for customization to meet local needs.

This capacity for expansion is particularly important in evolving land administration systems, where new laws, policies, and land management practices may require updates to the code lists over time. By utilizing this flexible code list structure, the LADM-based system ensures that any new classifications or regulations can be incorporated without requiring disruptive changes to the core database schema.

In conclusion, the combination of standardized code lists and the ability to extend them as needed ensures that the LADM model is not only robust and interoperable but also adaptable to the unique and changing requirements of different jurisdictions.

4.2.3. Table(database) for Class(UML)

In the process of converting UML classes to SQL database tables, each UML class is mapped to a corresponding database table. This transformation is a fundamental step in implementing the physical data model, ensuring that the attributes and relationships defined in the UML model are accurately reflected in the database schema. Typically, a UML class consists of several key components, such as class names, attributes, and cardinalities, which are then mapped to SQL table names, columns, and constraints respectively.

The accompanying Table 4.2 illustrates how the main components of a UML class translate into SQL database elements.

UML Component	SQL Representation	Example
Class Name	Table Name	UML Class: LA_Party → SQL Table: LA_Party
Attribute Name	Column Name	UML Attribute: name → SQL Column: name
Attribute Type	Data Type	UML Type: CharacterString → SQL Type: TEXT
Multiplicity: 1	NOT NULL Constraint	UML Multiplicity: 1 → SQL Constraint: name TEXT NOT NULL
Multiplicity: 0..1	Allow NULL (No constraint)	UML Multiplicity: 0..1 → SQL Column: name TEXT
Multiplicity: 0..*	Many-to-Many Relationship, Association Table	UML Multiplicity: 0..* → Requires a join table
Unique Identifier	PRIMARY KEY Constraint	UML Attribute: pID → SQL: pID INTEGER PRIMARY KEY
Inheritance (usually in italics in the upper right corner)	INHERITS keyword (for PostgreSQL)	UML: LA_Party inherits VersionedObject → SQL: INHERITS (VersionedObject)
Visibility (public/private/protected)	Ignored in SQL, all attributes accessible	N/A
Initial Value	DEFAULT Clause	UML availabilityStatus: LA AvailabilityStatusType = documentAvailable → SQL: availabilityStatus INTEGER DEFAULT 1 (the id of documentAvailable)

Table 4.2.: Mapping UML Components to SQL Representations

Example: Converting LA_Party to SQL Table

An example is the *LA_Party* class from the SDG 1.4.2 UML model, which represents parties involved in land administration as shown in Figure 4.1.

This class has the following attributes:

- `pID`: The object identifier (Oid) with cardinality 1, representing a non-null unique identifier for each party.
- `extPID`: An external party identifier with cardinality 0..*, representing a many-to-many relationship with external parties.
- `name`: A character string with cardinality 0..1, representing the party's name (optional).

- `humanSex`: Linked to a code list representing gender (LA_HumanSexesType), with cardinality 0..1 (optional).
- `type`: A required attribute representing the party type, such as naturalPerson or nonNaturalPerson.

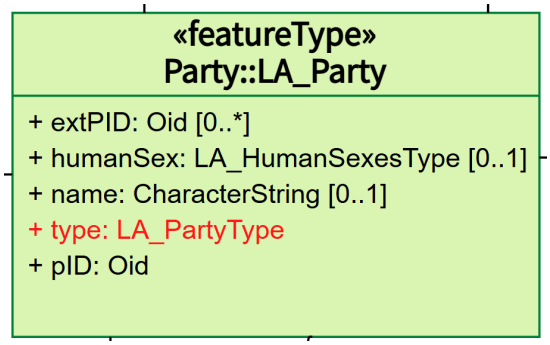


Figure 4.1.: LA_Party Class for SDG Indicator 1.4.2

Based on this, the UML class *LA_Party* can be transformed into the following SQL table structure:

```
CREATE TABLE LA_Party (
    pID INTEGER PRIMARY KEY,  -- Primary key representing the
       unique identifier of the party
    extPID INTEGER REFERENCES ExtParty(extPID),  -- Foreign key
       referencing ExtParty, representing many-to-many
       relationship
    name TEXT,  -- Optional attribute for the party's name
    humanSex INTEGER REFERENCES LA_HumanSexesType(hst_id),  --
       Optional foreign key referencing the gender code list
    p_type INTEGER NOT NULL REFERENCES LA_PartyType(pt_id)  --
       Required foreign key representing the party type
);
```

Each UML element is mapped to its corresponding SQL component to maintain the class's attributes and relationships:

- `pID`: In SQL, this is represented as `pID INTEGER PRIMARY KEY`, which enforces both uniqueness and the requirement for the value to be non-null.
- `extPID`: In the SQL schema, `extPID` is implemented as a foreign key (`REFERENCES ExtParty(extPID)`) linking to the *ExtParty* table. This captures the relationship between *LA_Party* and external parties (e.g., data from a census database). If a many-to-many relationship needs to be represented (i.e., where a party can be linked to multiple external identifiers such as `ext1_PID`, `ext2_PID`, etc.), an additional association table would be required to map the `pID` from *LA_Party* to multiple external party identifiers.
- `name`: The `name` attribute is simply implemented as a `TEXT` column, allowing `NULL` values since no additional constraints are specified.

- `humanSex`: In the SQL schema, this is represented by a foreign key referencing the *LA_HumanSexesType* table (`REFERENCES LA_HumanSexesType(hst_id)`). The absence of the `NOT NULL` constraint mirrors the optional nature of this attribute in UML.
- `p_type`: In SQL, it is implemented as a foreign key (`REFERENCES LA_PartyType(pt_id)`), linking to the *LA_PartyType* table. The `NOT NULL` constraint ensures that this attribute must always have a value, reflecting the UML class's requirements.

4.2.4. Relationships Between Tables

In a database schema derived from a UML model, the various relationships between tables are essential to accurately represent the structure and behaviour of the system. These relationships ensure data integrity and reflect the way different entities interact in the model. In this section, we will discuss the two main types of relationships used in LADM: inheritance and association.

Inheritance

Inheritance refers to a relationship where one class inherits the attributes of another class. In LADM, this structure is widely employed, and a typical example is the requirement that all land administration and georegulation systems must support bi-temporal data management. This is achieved through inheritance from the *VersionedObject* class (an example in Figure 4.2). The *VersionedObject* class provides several temporal attributes that ensure all classes inheriting from it can record and manage time-related data changes. These temporal attributes include the start and end of the lifecycle, as well as the real-world valid time.

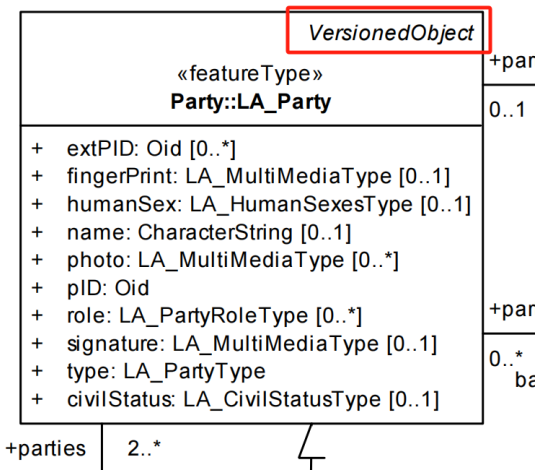


Figure 4.2.: Inheritance Relationships in Class *LA_Party*

LADM's bi-temporal system enables a distinction between system time (i.e., when data is inserted or modified in the database) and real-world time (i.e., the time at which the event occurred in reality). This mechanism is captured by the `beginLifespanVersion` and `endLifespanVersion` fields for system time, and by `beginRealWorldLifespanVersion` and

`endRealWorldLifespanVersion` for real-world events. Therefore, inheritance is crucial in LADM, as it enables the tracking and management of the system's state at any historical moment, which is essential for maintaining and auditing historical data.

In the UML model, inheritance is represented by lines with arrows (as shown in Figure 4.3). For instance, classes such as *LA_Party*, *LA_RRR*, and *LA_BAUnit* all inherit from *VersionedObject*, thereby acquiring bi-temporal properties.

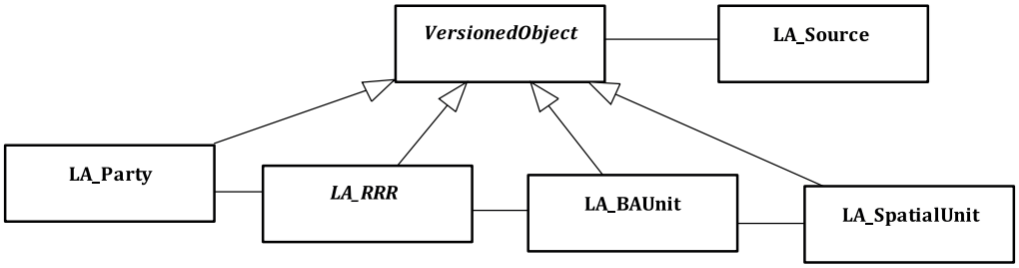


Figure 4.3.: Inheritance of Temporal Attributes from *VersionedObject* in LADM

The following SQL example defines the *VersionedObject* table, which captures the bi-temporal attributes:

```
CREATE TABLE VersionedObject (
    beginLifespanVersion TIMESTAMPTZ,
    endLifespanVersion TIMESTAMPTZ,
    beginRealWorldLifespanVersion TIMESTAMPTZ,
    endRealWorldLifespanVersion TIMESTAMPTZ,

    CONSTRAINT chk_version_lifespan CHECK (endLifespanVersion IS
        NULL OR endLifespanVersion > beginLifespanVersion),
    CONSTRAINT chk_version_realworld_lifespan CHECK (
       endRealWorldLifespanVersion IS NULL OR
       endRealWorldLifespanVersion >
       beginRealWorldLifespanVersion)
);
```

Classes such as *LA_Right* can inherit from this table, ensuring that they can utilize the bi-temporal system:

```
CREATE TABLE LA_Right (
    rID INTEGER PRIMARY KEY,
    r_type INTEGER NOT NULL REFERENCES LA_RightType (rt_id),

    -- inherited from LA_RRR
    suID INTEGER,
    pID INTEGER,
    share fraction,
    shareCheck BOOLEAN,
```

```
    timeSpec TEXT
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

The provided SQL code demonstrates two distinct forms of inheritance: temporal attribute inheritance from the *VersionedObject* class and attribute inheritance from the *LA_RRR* class, but these two types of inheritance are handled differently in SQL due to their scope and applicability.

In this example, *LA_Right* inherits the temporal attributes from *VersionedObject* using the SQL `INHERITS` keyword. This approach is appropriate because bi-temporal attributes are required across many different classes in LADM (except *LA_Source* and its subclasses), making it efficient to centralize them in a parent table. By doing so, classes like *LA_Party*, *LA_SpatialUnit*, and *LA_Right* can all share the same set of temporal attributes without redundant code.

On the other hand, *LA_Right* also inherits attributes from the*LA_RRR* class, such as `suID`, `pID`, `share`, `shareCheck`, and `timeSpec`. However, this inheritance is handled differently. Since in the SDG 1.4.2 UML diagram only the *LA_Right* class inherits from *LA_RRR*, it is more efficient in SQL to replicate these attributes directly in the *LA_Right* table. This avoids the overhead of creating a separate table for *LA_RRR* when only one class requires these attributes. Therefore, the attributes are explicitly written in the *LA_Right* table instead of using table inheritance.

This distinction between temporal inheritance and attribute inheritance is important in designing efficient databases. Temporal attributes are widely shared across the model, justifying the use of table inheritance to avoid redundancy. For more specific cases, such as *LA_Right* inheriting from *LA_RRR*, direct inclusion of the attributes in the table is more appropriate due to the limited scope of the inheritance.

Association

Association represents a relationship between two or more classes, indicating how instances of these classes are related to each other. In the context of LADM, associations between classes capture important connections, such as land rights being held by parties or land units being linked to specific rights and responsibilities. These associations are critical for expressing the relationships between different land administration entities, allowing for the modeling of complex interactions in the land tenure system.

In UML diagrams, associations are typically represented by a line connecting two classes, with multiplicities at each end, indicating the number of instances involved in the relationship. For instance, in the UML diagram for SDG 1.4.2, the *LA_Party* class is associated with the *LA_Right* class, which captures the idea that parties hold specific rights over land. The multiplicity of this association might indicate, for example, that a party can hold one or more land rights, and that a land right must be associated with exactly one party as shown in Figure 4.4.

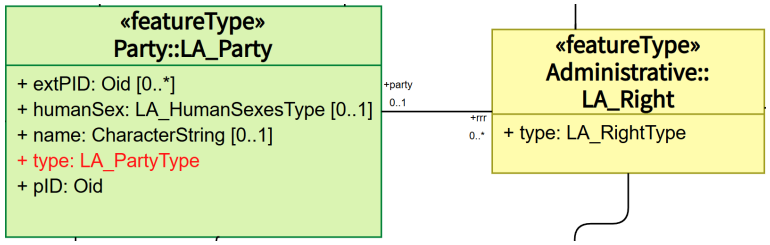


Figure 4.4.: Association Relationships Between Class LA_Party and LA_Right

In SQL, associations are usually implemented through foreign keys. A foreign key is a field (or collection of fields) in one table that uniquely identifies a row in another table. This creates a direct link between two tables, ensuring that the associated entities remain connected and their relationships are maintained. For example, the association between *LA_Party* and *LA_Right* is represented by adding a foreign key in the *LA_Right* table, referencing the primary key of the *LA_Party* table.

The following example shows how an association between *LA_Party* and *LA_Right* can be implemented in SQL:

```
CREATE TABLE LA_Party (
    pID INTEGER PRIMARY KEY,  -- Primary key of LA_Party
    -- rest of other attributes
);

CREATE TABLE LA_Right (
    rID INTEGER PRIMARY KEY,              -- Primary key of
       LA_Right
    pID INTEGER REFERENCES la_party(pid) -- Foreign key
       referencing LA_Party
    -- rest of other attributes
);
```

In this example, the `pID` attribute in *LA_Right* serves as a foreign key, linking each record in *LA_Right* to a corresponding record in *LA_Party*. This enforces the association between the two tables, ensuring that each land right is held by a valid party. The foreign key constraint maintains referential integrity, ensuring that only valid `pID` values from *LA_Party* can be entered in the *LA_Right* table.

The multiplicity of the association in UML (e.g., 1..* or 0..1) can also influence how the foreign keys are implemented. For instance, if the association is one-to-many (i.e., one party can hold many rights), a foreign key in *LA_Right* referencing *LA_Party* is sufficient. However, when the association is many-to-many, such as the relationship between `extaddress` and *LA_SpatialUnit* in the context of SDG 1.4.2, an additional join table is required. In this case, multiple `extaddress` instances can be associated with multiple *LA_SpatialUnit* instances. This is because an `extaddress` can be categorized in various ways, such as by region or urban/rural classification, while a *LA_SpatialUnit* can correspond to multiple `extaddress` instances. To represent this many-to-many relationship, a join table, *extaddress_suid_relation*, is used to store the associations, as shown below:

```
CREATE TABLE extaddress_suid_relation (
    extaddressid INTEGER NOT NULL,
    suid INTEGER NOT NULL,
    suid_geom geometry NOT NULL,
    PRIMARY KEY (extaddressid, suid),
    FOREIGN KEY (suid) REFERENCES la_spatialunit(suid)
);
```

This table maintains the many-to-many relationship between `extaddress` and *LA_SpatialUnit*, ensuring that each extaddress can be linked to multiple spatial units and vice versa.

4.2.5. Constraints

In database design, constraints play a critical role in ensuring data integrity and consistency. They restrict data entry or updates to ensure that the information stored in the database adheres to predefined rules. Some constraints have already been mentioned in previous sections, this section will provide a comprehensive summary of the various types of constraints used in the database model for SDG 1.4.2.

Primary Key

In database design, primary keys are not only used to uniquely identify each record but also reflect the uniqueness constraints between entities as depicted in UML diagrams. Each instance of a class in a UML diagram requires a unique identifier, which is enforced through primary keys in the physical database. A primary key ensures that each record in the table has a unique identifier that cannot be duplicated or set to `NULL`. For example, in the *LA_Party* table, `pID` serves as the primary key, uniquely identifying each party entity. The syntax is straightforward, with the primary key constraint added directly to the relevant column.

```
CREATE TABLE LA_Party (
    pID INTEGER PRIMARY KEY
    -- Other attributes
);
```

Foreign Key

In database design, a foreign key enforces relationships between tables, ensuring that a field in one table corresponds to the primary key of another table. This constraint guarantees referential integrity, meaning that relationships between records are valid and consistent. Foreign keys help establish links between entities, enforcing the logical connections defined in the conceptual model.

In a UML diagram, foreign key relationships are depicted as associations between different classes. These associations represent connections where one entity is related to another. The association between classes in the UML diagram is mapped to a foreign key in the physical database, where one class (entity) references the primary key of another.

In SQL, foreign keys are explicitly defined using the `FOREIGN KEY` constraint, which links a column in one table to the primary key in another table. This ensures that the value in the foreign key column always matches a valid record in the referenced table. For example, in the *LA_PartyMember* table, `pID` is a foreign key that references the `pID` in the *LA_Party* table, establishing a relationship between party members and parties:

```
CREATE TABLE LA_PartyMember (
    pmID INTEGER PRIMARY KEY,
    pID INTEGER REFERENCES LA_Party(pID), -- Foreign key linking
        to the LA_Party table
    pgID INTEGER REFERENCES LA_PartyGroup(groupID) -- Foreign
       key linking to the LA_PartyGroup table
    -- Other attributes
);
```

Alternatively, foreign keys can be defined separately after all columns are listed. For example:

```
CREATE TABLE LA_PartyMember (
    mpID INTEGER PRIMARY KEY,
    pID INTEGER, -- Foreign key defined separately
    pgID INTEGER, -- Foreign key defined separately share
       NUMERIC,
    FOREIGN KEY (pID) REFERENCES LA_Party(pID), -- Separate
       foreign key definition
    FOREIGN KEY (pgID) REFERENCES LA_PartyGroup(groupID) --
       Separate foreign key definition
);
```

Other Types of Constraints

In addition to primary and foreign keys, other constraints in database design ensure data integrity and enforce specific rules. These include:

1. **CHECK**
   The `CHECK` constraint ensures that values in columns satisfy specific conditions. For example, in the *VersionedObject* table, to avoid generating invalid time intervals, rules exist: the end time must be later than the start time, or the end time must be null (i.e., the object still exists or the life cycle has not ended).

   ```
   CREATE TABLE VersionedObject (
           beginLifespanVersion TIMESTAMPTZ,
           endLifespanVersion TIMESTAMPTZ,
           beginRealWorldLifespanVersion TIMESTAMPTZ,
           endRealWorldLifespanVersion TIMESTAMPTZ,

           CONSTRAINT chk_version_lifespan CHECK (
              endLifespanVersion IS NULL OR endLifespanVersion
              > beginLifespanVersion),
   ```

```
        CONSTRAINT chk_version_RealWorldLifespan CHECK (
            endRealWorldLifespanVersion IS NULL OR
            endRealWorldLifespanVersion >
            beginRealWorldLifespanVersion)
);
```

2. **UNIQUE and NOT NULL**
   The `UNIQUE` and `NOT NULL` constraints are often used together to ensure both the uniqueness and mandatory nature of certain attributes. In UML, this often corresponds to the description field in code lists, which must be distinct and not empty. For instance, in the *LA_RightType* table:

```
CREATE TABLE LA_RightType (
    rt_id SERIAL PRIMARY KEY,
    rt_description TEXT NOT NULL UNIQUE
);
```

4.3. Functions and Triggers for Complex Constraints

In previous sections, simple constraints (such as primary key, foreign key, and check constraints) were introduced, and they are usually sufficient to enforce basic rules, like uniqueness or nullability of fields. However, in complex systems like the Land Administration Domain Model, there are some rules and relationships that cannot be fully enforced by basic constraints alone. This is where complex constraints come into play.

Complex constraints are rules that go beyond simple field-level validation and require advanced logic to ensure data integrity across multiple records, tables, or transactions. For example, versioning is an important part of tracking the lifecycle of land rights. This requires complex constraints to ensure that the lifecycle of each version of a record is continuous, with no gaps or overlaps in time. The complex constraints in land tenure change management ensure the accuracy and long-term validity of land tenure records.

PostgreSQL supports the enforcement of such complex constraints through the use of `functions` and `triggers`. Functions can encapsulate the logic required to enforce advanced validation rules. Triggers are mechanisms that automatically invoke these functions before or after certain database operations, such as inserting or updating records. This combination of functions and triggers allows for the dynamic validation of complex constraints during database transactions, ensuring that the integrity of the data is maintained in accordance with the LADM's specific business rules.

Below are examples of how complex constraints are implemented in practice.

4.3.1. check_version_lifespan_continuity

The `check_version_lifespan_continuity` constraint ensures that object versions in the system adhere to the temporal consistency requirements defined by LADM. Each version of a record must have clearly defined start and end times, and for any given entity, the end

time of the previous version should match the start time of the next version. This ensures continuity and prevents gaps in the records, preserving the historical integrity of the data.

In the UML class diagram for LADM, the *VersionedObject* class contains the attributes `beginLifespanVersion` and `endLifespanVersion`, which represent the system timestamps for when a record begins and ends. The constraint that governs the continuity of these versions is often represented in the UML diagram using a constraint note, as shown in Figure 4.5. This constraint ensures that the `endLifespanVersion` of one record (n-1) must match the `startLifespanVersion` of the next record (n).

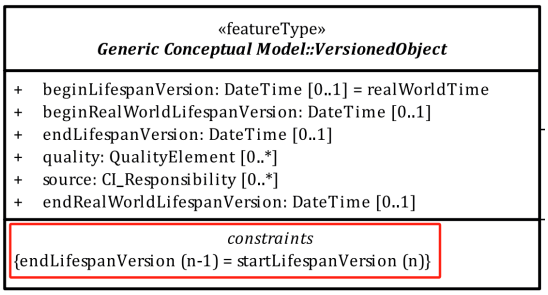


Figure 4.5.: *VersionedObject* UML Diagram with Lifespan Continuity Constraint

In SQL, the `check_version_lifespan_continuity` constraint is implemented through a trigger function to ensure temporal continuity between records. This trigger function is automatically triggered when a new record is inserted or an existing record is updated, and checks whether the `beginLifespanVersion` of the new record matches the `endLifespanVersion` of the previous version.

When a new record is inserted or an existing record is updated, the trigger checks if the `beginLifespanVersion` is non-null and if the `endLifespanVersion` of the previous record matches this value. If there is a discontinuity (i.e., the `beginLifespanVersion` of the new record does not match the `endLifespanVersion` of the previous version), the function raises an exception, preventing the insertion or update of the record. This approach ensures that the system maintains a continuous and accurate historical record, preventing any gaps or inconsistencies in the versioned data.

The function dynamically generates an SQL query to check the continuity between records, using the primary key of the record to identify the corresponding version and ensure temporal consistency. By raising exceptions when discrepancies are found, the system prevents the introduction of invalid temporal data.

In SQL, the `check_version_lifespan_continuity` constraint is enforced through a trigger function, ensuring the continuity of versioned data. The logic of this implementation is as follows:

- When a new record is inserted or an existing record is updated, the trigger function dynamically generates an SQL query to check if the current record's `beginLifespanVersion` is not null and whether the previous version's `endLifespanVersion` matches the current record's `beginLifespanVersion`.

- If the continuity check fails, the function raises an exception, preventing the data from being inserted or updated, thus maintaining the integrity and consistency of the versioned data.

The SQL implementation consists of several key steps:

1. First, the function extracts the primary key value of the current record and generates an SQL query using the `format` function to check if the previous version's `endLifespanVersion` matches the current version's `beginLifespanVersion`.
2. If the match fails, an exception is raised using `RAISE EXCEPTION`, specifying which record's lifecycle continuity is problematic.
3. If the new record is an initial insertion (i.e., `endLifespanVersion` is null), no version continuity check is performed, and the record is directly returned.

To automate the enforcement of this constraint, triggers are created for each table inheriting from *VersionedObject*. These triggers ensure the function is executed automatically before any insert or update operation, maintaining the temporal consistency of the data. The full code and trigger implementation are provided in the appendix.

4.3.2. check_fraction_validity

In LADM, multiple parties can share rights to the same parcel of land. In such cases, it is essential to ensure that the share distribution among parties is fair and legally valid. Therefore, as discussed in Section 4.2.1, the Fraction data type is subject to several constraints to guarantee the validity of the share values. These constraints are clearly defined in the UML model, as illustrated in the figure 4.6

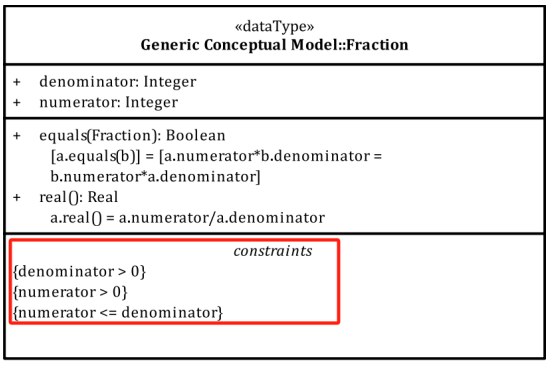


Figure 4.6.: Fraction Data Type Constraints in UML diagram

Specifically:

- **The denominator must be greater than zero**: This ensures that the lower bound of the fraction is positive, which is a fundamental requirement for a valid fraction.
- **The numerator must be non-negative**: The numerator cannot be less than zero to prevent negative shares from being assigned.

- **The numerator must not exceed the denominator**: The value of the numerator cannot be greater than the denominator to avoid having a fraction that exceeds 1.

In the SQL implementation, a series of conditional statements (`IF...THEN`) are used within the trigger function to validate whether the fraction values conform to the constraints defined in the LADM model. The approach involves the following steps:

- The first check ensures that the share field is not `NULL`. If it is, an exception is raised using the statement `IF NEW.share IS NULL THEN`.
- The second check, `IF (NEW.share).denominator` $\leq$ `0 THEN`, ensures that the denominator is greater than zero.
- The next condition, `ELSIF (NEW.share).numerator` $<$ `0 THEN`, validates that the numerator is non-negative, preventing the storage of invalid negative shares.
- Finally, the condition `ELSIF (NEW.share).numerator` $>$ `(NEW.share).denominator THEN` ensures that the numerator does not exceed the denominator.

4.3.3. check_share_sum

The `check_share_sum` constraint ensures that when multiple parties hold rights to the same land unit, the total sum of these rights must equal 1, preventing errors such as a party holding more rights than allowed. This constraint applies to the share attribute in the *LA_Right* class, as shown in the Figure 4.7. The `shareCheck` attribute indicates whether this sum constraint is enforced for a specific type of right. When `shareCheck` is set to true, the system must verify that the total sum of rights equals 1; otherwise, the check is skipped. In the UML model, this constraint is expressed as an invariant, meaning it applies to all instances where `shareCheck` is enabled.

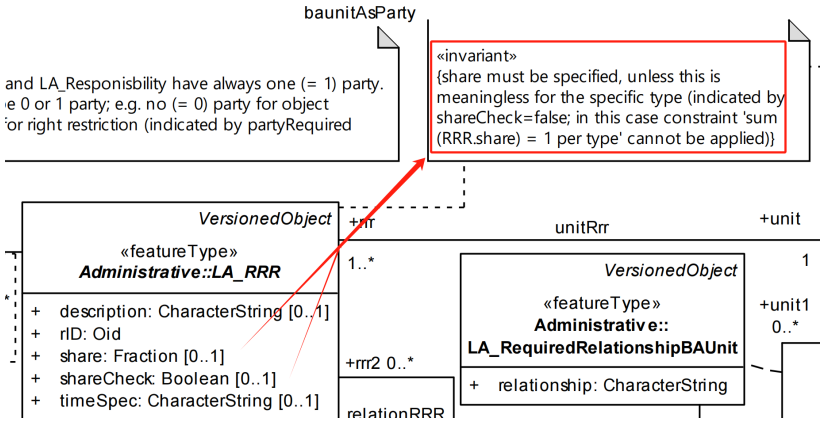


Figure 4.7.: Share Sum Constraint of *LA_Right* in UML diagram

In the SQL implementation, the `check_share_sum` function is used to verify that the total share for all *LA_Right* records equals 1, provided that the `shareCheck` is set to true. The logic is as follows:

1. **Condition check**: First, the function checks if `shareCheck` is true or NULL. If `shareCheck` is false, no further checks are performed, and the new or updated record is returned directly. This ensures that the share constraint is only applied when necessary.

2. **Query construction**: If `shareCheck` is true, the function dynamically constructs a query to retrieve all relevant share values from the *LA_Right* table, filtering by spatial unit (`suid`), right type (`r_type`), and `beginRealWorldLifespanVersion`. This ensures that only records pertaining to the same spatial unit and right type are considered.

3. **Summation**: The function then iterates through the query results, converting all shares to a uniform denominator to facilitate summation. This ensures that the shares from different records can be accurately combined.

4. **Validation**: After adding the share of the current record to the total, the function checks if the sum equals 1. If the sum does not equal 1, an exception is raised, preventing the insertion or update of the record.

4.3.4. check_minimum_group_members

This function enforces a constraint to ensure that each *PartyGroup* must have at least two members, as indicated by the multiplicity of 2..* between *LA_Party* and *LA_GroupParty* in Figure 4.8. This constraint is critical for Group Parties, as any *PartyGroup* must consist of at least two individuals; otherwise, the group would not qualify as a valid entity in land administration.

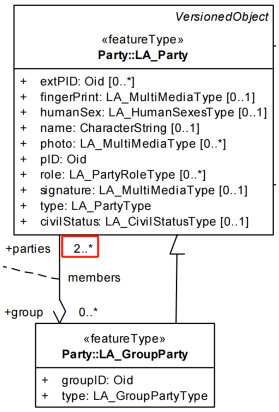


Figure 4.8.: Multiplicity constraints between *LA_Party* and *LA_GroupParty* in UML

In the UML model, this constraint is represented by the 2..* multiplicity on the association line between *LA_Party* and *LA_GroupParty*. This means that each group must contain at least two members.

Although this constraint applies to the relationship between *LA_Party* and *LA_GroupParty*, the actual constraint check is performed on the *LA_PartyMember* table in SQL. This is because the records of group members are stored in the *LA_PartyMember* table, which represents the individual members within a group. The trigger is also created on the *LA_PartyMember* table,

ensuring that each time a member is added or removed from a PartyGroup, the check for the minimum number of members is automatically executed.

In SQL, the `check_minimum_group_members` function verifies whether this condition is met by counting the number of members associated with a given *PartyGroup* in the *LA_PartyMember* table. Specifically, it checks the `pgID` (PartyGroup ID) to match the newly inserted or updated group record. The function raises an exception if the member count is less than two.

4.3.5. check_administrative_source_constraints

The `chec_administrative_source_constraints` function ensures that the data in the *LA_AdministrativeSource* remains consistent with the corresponding records in the *LA_Right*. This constraint is critical because each *VersionedObject* must maintain temporal consistency with its corresponding *LA_Source*, as shown in Figure 4.9. Specifically for this function, each administrative source (*LA_AdministrativeSource*) must align with the legal rights (*LA_Right*) in land administration. The function verifies that the lifecycle timestamps and acceptance dates between the two tables are synchronized during insert or update operations.

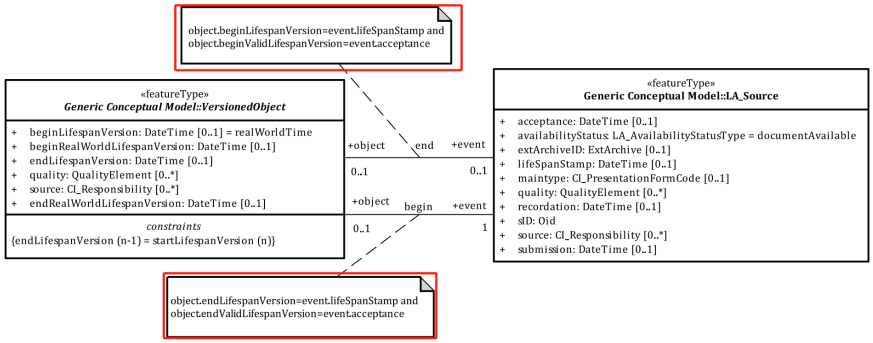


Figure 4.9.: Constraints for *VersionedObject* and *LA_Source* in UML

In the SQL implementation:

1. **Matching Administrative Source and Right Records:** The function begins by checking if a record with the same asid already exists in the *LA_AdministrativeSource* table. This helps to differentiate between insert and update operations. It also queries the *LA_Right* table for the corresponding `rID` to ensure that every administrative source is tied to a valid right record.
2. **Timestamp Validation:**
   - **For Acceptance Date:** The `acceptance` field in *LA_AdministrativeSource* must match the `beginRealWorldLifespanVersion` in *LA_Right*. This ensures that the administrative source reflects the correct real-world starting point of the associated right.
   - **For Lifespan Timestamps:** Depending on whether the operation is an insert or an update, different `timestamps` are validated:
     - **Insert Operation:** The `lifeSpanStamp` in *LA_AdministrativeSource* must match the `beginLifespanVersion` in *LA_Right*, ensuring that the source document is aligned with the start of the right's system lifespan.

– **Update Operation:** The `lifeSpanStamp` must match the `endLifespanVersion` in *LA_Right*, ensuring that any updates to the source are correctly synchronized with the end of the right's system lifespan.

4.4. Functions for Calculation

4.4.1. countAdult

According to the SDG 1.4.2 metadata, in Definition section, the adult population of a country is measured using census data or surveys based on an adequate sampling frame. In terms of data sources, the metadata identifies census data as a primary source, though it includes individuals of all age groups. Therefore, accurately filtering and counting the adult population is essential for subsequent calculations. This forms the basis of the `countAdult` function. In the methodology section of SDG 1.4.2, it is stated that the indicator is composed of two parts: Part (A) measures the proportion of adults with legally recognized documentation of land; Part (B) measures the proportion of adults who perceive their land rights as secure. Both parts rely on accurate adult population data, making the selection and counting of adults a crucial step.

In the UML model pseudocode, as illustrated in Figure 4.10, the `countAdult` function operates by calling the `getPeopleInArea` function to retrieve all individuals within a specified geographic area. It then compares their birthdates with a given reference date to calculate their age. If an individual is 18 years or older, they are counted as part of the adult population. This logic iterates over each individual, calculates their age, and ultimately returns the total count of adults that meet the criteria.

```
《method》
// pseudocode
public int countAdults(int year, Geometry area) {
   int adultCount = 0;
   Date currentDate = new Date(year, 1, 1);
   List<Person> peopleInArea = getPeopleInArea(area);

   for (Person person : peopleInArea) {
      Date birthday = person.getBirthday();
      int age = currentDate.getYear() - birthday.getYear();
      if ((birthday.getMonth() > currentDate.getMonth()) ||
         (birthday.getMonth() == currentDate.getMonth() && birthday.getDay() >
currentDate.getDay())) {
         age--;
      }
      if (age >= 18) {
         adultCount++;
      }
   }
   return adultCount;
}

private List<Person> getPeopleInArea(Geometry area) {
   return peopleInArea; //spatial queries
}
```

Figure 4.10.: Pseudocode of Method `countAdult` in UML

The SQL implementation of this logic is more detailed and structured to ensure geographical consistency, accurate age calculation, time validity, and deduplication of individual records.

- **Input Parameters**: `CA_begindate` (date type) to specify the reference date for calculating age, and `CA_area` (geometry type) representing the geographic area.
- **Geographical Filtering**: This function is handled by the PostGIS function `ST_Contains`, which ensures that only individuals located within the specified geographic region (CA_area) are selected. By using the `suid_geom` geometry field from the `extaddress_suid_relation` table, the function matches individuals' addresses to the specified geographic area. The condition `ST_Contains(CA_area, esr.suid_geom)` ensures that only individuals within the defined spatial boundaries are considered in the count.
- **Age Calculation**: Use PostgreSQL's `age` function, combined with `EXTRACT(YEAR FROM age(...))` to extract the age of each individual. Specifically, `age(CA_begindate, e.birthday)` calculates the age of an individual on the specified `CA_begindate`. The condition `EXTRACT(YEAR FROM age(...))>= 18` ensures that only individuals who are 18 years or older on the given date are included. This logic guarantees that the final count accurately reflects the adult population.
- **Time Validity Checks**: Use the `begindate` and `enddate` fields to ensure that the records are valid on the specified reference date. The condition `e.begindate <= CA_begindate` ensures that the record for each individual was valid by the reference date, while `(e.enddate IS NULL OR e.enddate >= CA_begindate)` ensures that the record had not ended by that date or is still ongoing (indicated by a `NULL enddate`).
- **Data de-duplication and counting**: Use `DISTINCT ON (e.extpid)` to ensure that each individual (identified by `extpid`) is counted only once. This deduplication process is based on the unique individual identifier (`extpid`), ensuring that each person is included in the count only once, even if they appear multiple times in the dataset. The total number of adults meeting these conditions is then computed using `COUNT(*)` and returned.

This comprehensive SQL implementation ensures that the `countAdult` function accurately filters and counts the adult population, providing a reliable foundation for further computations. The full code is provided in the appendix.

4.4.2. computeProportionWithLegalDocumentation

According to the metadata of SDG 1.4.2, part A focuses on calculating the proportion of the adult population that holds legally recognized land documentation. The metadata defines "Legally recognized documentation" as government-recognized legal documents related to land rights, such as title deeds or lease agreements. The specific types of legally recognized land documentation vary by country, as they depend on national legal regulations. The metadata also indicates that administrative data reported by national land agencies, often from land registries or cadastral systems, can be used for this calculation. These systems provide information on adults granted legal rights to land, along with details about the location of these rights. When these systems are fully electronic, the data can be updated annually.

```
《method》
// pseudocode
public float computeProportionWithLegalDocumentation(int year, Geometry area, LA_HumanSexesType gender, LA_RightType
tenureType) {
    GenderTenureKey key = new GenderTenureKey(gender, tenureType);
    int adultsWithDocumentationCount = 0;
    if (adultsWithLegalDocumentation.containsKey(key)) {
        adultsWithDocumentationCount = adultsWithLegalDocumentation.get(key);
    }
    int totalAdultsCount = countAdults(year, area);

    if (totalAdultsCount > 0) {
        return (float) adultsWithDocumentationCount / totalAdultsCount;
    } else {
        return 0.0f;
    }
}

public float computeProportionPerceivingSecurity(int year, Geometry area, LA_HumanSexesType gender, LA_RightType
tenureType) {
    // Similar logic
        return (float) adultsPerceivingSecurityCount / totalAdultsCount;
}
```

Figure 4.11.: Pseudocode of Method computeProportionWithLegalDocumentation in UML

The UML model's `computeProportionWithLegalDocumentation` function is illustrated in Figure 4.11, showing its pseudocode logic. First, the function calls the `countAdults` function to determine the total adult population in a specific geographic area. It then filters for individuals who possess legally recognized land documentation. The computation is categorized by gender and land tenure type, associating these classifications with records through a GenderTenureKey. In the pseudocode, if a record matches the classification key, the function tallies the number of adults with legal documentation. Finally, the function calculates the proportion by dividing the number of adults with legal documentation by the total adult population.

In the SQL implementation, the `computeProportionWithLegalDocumentation` function is designed to combine geospatial filtering (using PostGIS) with demographic filtering to compute the proportion of adults holding legally recognized land documentation within a specific area and time frame. The design logic is as follows:

1. **Input Parameter**s: To allow analysis for any area or time period, the function accepts three parameters: a start date (`input_begin`), an end date (`input_end`), and a geometric object (`input_area`) representing the geographic boundary. PostGIS's `GEOMETRY` type enables spatial operations, ensuring that only individuals located within the specified area are counted.
2. **Return Type**: The function returns a table with the following fields:
    - `category`: Indicates the type of classification (e.g., total population, gender, or tenure type).
    - `subcategory`: Specifies the specific subclass (e.g., male or female, but formatted as the corresponding number in the codelist).
    - `proportion`: Represents the proportion of adults holding legal land documentation.

3. **Total Population Proportion Calculation**: The function first calls the `countAdults` function to calculate the total number of adults within the specified geographic area and time range. If the total number of adults is greater than zero, the function proceeds to calculate the number of adults holding valid land ownership documents. Specifically, this is done by joining multiple tables (shown in Figure 4.12) and filtering the data based on certain conditions. The filtering criteria ensure that the individual's address is within the specified geographic area (`ST_Contains`), their records are valid within the specified time range (`beginLifespanVersion` and `endLifespanVersion`), and they are at least 18 years old on the reference date (calculated using the birthday field). Additionally, the function ensures that the associated administrative document is valid, meaning that the `as_type`, `pid`, and `suid` fields in the *la_administrativesource* table are not null, thus identifying adults who hold valid land ownership documents.

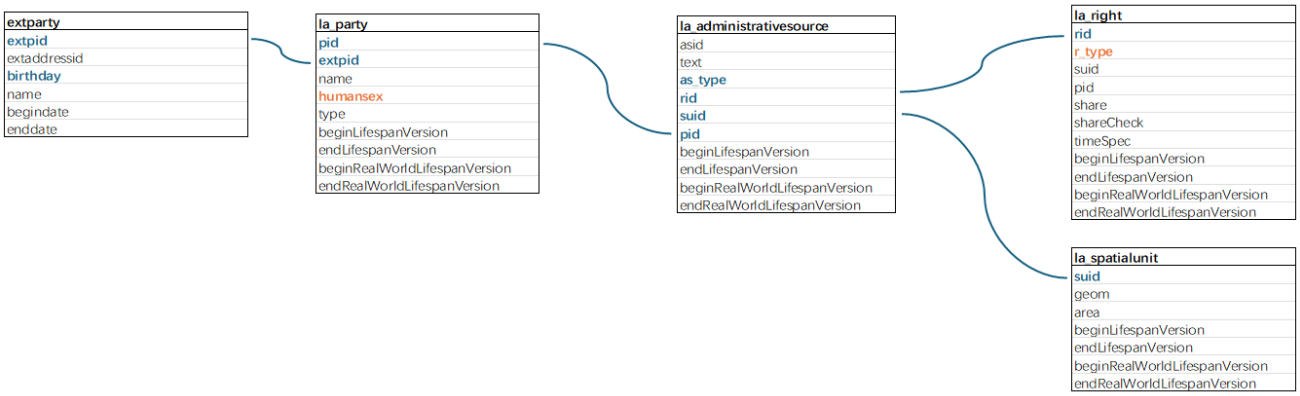


Figure 4.12.: Tables joined diagram for SDG 1.4.2(a)

```
totalAdultsCount := countAdults(input_begin::DATE,
    input_area);
IF totalAdultsCount > 0 THEN
     SELECT COUNT(*) INTO adultsWithDocumentationCount
     FROM ( ... ) AS unique_adultsWithDocumentationCount;
```

The function computes the proportion of the total adult population that holds legal documentation by performing a simple division, which is returned as the first result.

4. **Gender Categorization**: To further analyze the data, the function classifies adults by gender. It queries the *LA_Party* table's `humansex` field to retrieve the list of genders. For each gender group, the function applies similar logic to that used for the total population, calculating the proportion of individuals in each gender group who hold legal documentation.

```
FOR gender IN SELECT DISTINCT humansex FROM la_party LOOP
     SELECT COUNT(*) INTO adultsWithDocumentationCount
     FROM ( ... ) AS unique_adultsWithDocumentationCount;
     RETURN QUERY SELECT 'gender' AS category, gender::TEXT,
                                 adultsWithDocumentationCount
                                    ::FLOAT /
```

```
                                        totalAdultsCount::FLOAT
                                        AS proportion;
END LOOP;
```

5. **Tenure Type Categorization**: Following the gender classification, the function categorizes the adult population by land tenure type. This categorization is based on the r_type field in the *LA_Right* table, which indicates the type of land tenure held by an individual. By applying this classification, the function can analyze the distribution of legal documentation across different tenure types.

```
FOR tenureType IN SELECT DISTINCT r_type FROM la_right LOOP
    SELECT COUNT(*) INTO adultsWithDocumentationCount
    FROM ( ... ) AS unique_adultsWithDocumentationCount;
    RETURN QUERY SELECT 'tenure_type' AS category,
       tenureType::TEXT,
                                    adultsWithDocumentationCount
                                        ::FLOAT /
                                        totalAdultsCount::FLOAT
                                        AS proportion;
END LOOP;
```

6. **Output Results**: The final output of the function is a table containing three classifications (total population, gender, and tenure type) and the corresponding proportion for each subclass. It is worth noting that while the proportions output by this code are suitable for use in SDG reporting, in practice they are modified in small parts to include counts (e.g., total adults or adults with legal documents) for validation purposes by comparing the system output with the results of manual calculations.

computeProportionPerceivingSecurity

Similar to the computation of the proportion of adults with legal land documentation, the `computeProportionPerceivingSecurity` function calculates the proportion of adults who perceive their land rights as secure. This corresponds to Part B of SDG 1.4.2, which aims to measure the subjective perception of land tenure security among the adult population.

In the UML model, the `computeProportionPerceivingSecurity` function follows a similar logic to the `computeProportionWithLegalDocumentation` function. However, instead of filtering for adults with legal documentation(in *la_administrativesource*), it filters based on those adults who, according to survey data, perceive their land rights as secure(in *extsecurelandrightsquestionnaire*). Like before, it first calls the `countAdults` function to obtain the total adult population in a specific geographic area. Then, it filters the data based on gender and tenure type to identify those who perceive their land rights as secure.

The SQL implementation also follows a similar approach to the legal documentation function, with the main difference being the tables that are joined (as shown in Figure 4.13). Since the overall logic remains consistent, it will not be repeated here, and the full implementation can be found in the appendix. The final output is also similar to that of `computeProportionWithLegalDocumentation`, providing three main classifications (total adult population, gender, and tenure type) and returning the proportion or count of adults who perceive their land rights as secure within each subclass.

Figure 4.13.: Tables joined diagram for SDG 1.4.2(b)

4.5. Views and Report Generation

In databases, a view is a virtual table generated by querying and combining data from multiple tables. For SDG 1.4.2 reporting, the `SDG_1_4_2_Report` function produces detailed data on land documentation and tenure security, while views enhance data readability by mapping coded values to descriptive labels. Views allow users to quickly access formatted and aggregated information without directly modifying the underlying data, providing strong support for decision-making and analysis within land management systems.

4.5.1. SDG_1_4_2_Report

The design rationale behind the `SDG_1_4_2_Report` function is to simplify the process of generating reports for SDG 1.4.2 indicators. This function efficiently integrates previously developed calculation functions, reducing the complexity of manual intervention. It allows users to automatically generate comprehensive reports by simply inputting a time range and geographical area. This eliminates the need for users to call individual functions separately, thus improving the efficiency and accuracy of report generation while helping land administration systems more effectively meet the requirements of SDG 1.4.2.

In the UML model, the design logic of the `SDG_1_4_2_Report` function demonstrates how to generate a comprehensive report that analyzes the proportion of adults holding legally recognized land documentation and their perceived tenure security in a specific time period and geographical area. The function calculates the relevant proportions by iterating over all combinations of gender and land tenure types, ensuring all possible classifications are covered. As shown in Figure 4.14, the pseudocode illustrates the core steps of the function, which include initializing the report object, calculating the proportions of adults with legal documentation and perceived tenure security, and recording the results in the report. By calling the computeProportionWithLegalDocumentation and computeProportionPerceivingSecurity functions, the function can generate detailed analytical results for each gender and land tenure type combination. This modular design makes the function easy to expand and maintain, ensuring it can quickly adapt to additional data dimensions in the future.

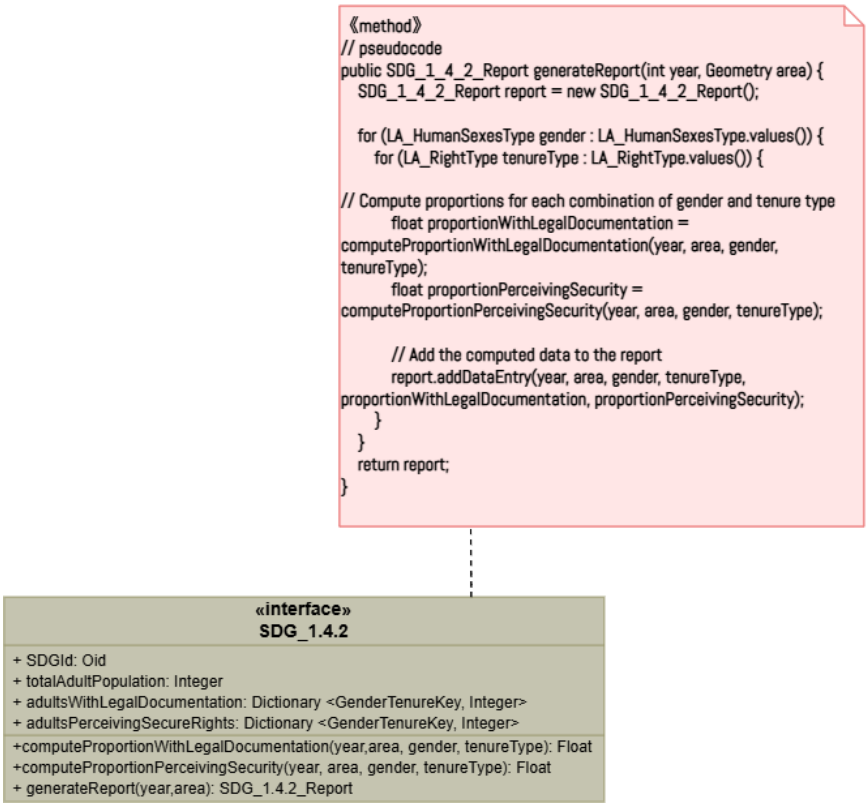


Figure 4.14.: Pseudocode of Method `SDG_1_4_2_Report` in UML

In SQL, the main implementation steps of the `SDG_1_4_2_Report` function align with the pseudocode structure in the UML model. Its design logic involves transforming the input geographical area using the `ST_Transform` function to ensure spatial consistency. Then, the `countAdults` function is called to calculate the total number of adults in the specified area, serving as the baseline for subsequent proportion calculations.

The data returned by the function is structured into a table containing fields for time, area, category, subcategory, and two types of proportions (proportion of adults with legal documentation and proportion of adults perceiving security). The entire calculation process is divided into three parts: first, calculating the overall population proportion; second, dividing the population by gender and calculating the proportion of adults within each gender group who hold legal land documentation and perceive security; and finally, classifying the population by land tenure type and calculating the corresponding proportions for each tenure type.

At each stage, the proportions are computed via subqueries that call two independent calculation functions, computeProportionWithLegalDocumentation and computeProportionPerceivingSecurity, which compute the proportion of adults holding legal documentation and the proportion perceiving tenure security, respectively. Through nested queries and grouping operations, the function outputs detailed proportions for each category and subcategory of the adult population. The final report provides multidimensional data, covering overall,

gender-based, and land tenure type-based statistics, offering detailed data support for land tenure analysis.

4.5.2. Create View

The creation of views is essential for transforming the raw data output of functions like `SDG_1_4_2_Report` into a structured and readable format, conducive to the reporting requirements of SDG 1.4.2 indicators. The raw output from these functions is typically populated with coded values, such as identifiers for gender or tenure type. These codes, while efficient for data processing, are not user-friendly for stakeholders or policymakers who require detailed and understandable reports. Views are implemented to merge the output from `SDG_1_4_2_Report` with relevant codelists, which contain descriptive labels that replace the coded values with more interpretable terms. This method significantly enhances the clarity and usability of the data by providing stakeholders with a report that is both informative and accessible.

The following example demonstrates the process of creating a view using data from the `SDG_1_4_2_Report` function:

```
CREATE OR REPLACE VIEW SDG_1_4_2_Report_View_2000_all AS
WITH report_data AS (
    SELECT *
    FROM SDG_1_4_2_Report('2000-01-01 00:00:00', '2001-01-01
       00:00:00', (SELECT geom FROM test_area WHERE id = 1))
)
SELECT
    report_begindate,
    report_enddate,
    report_region,
    report_category,
    CASE
        WHEN report_category = 'gender' THEN hst.hst_description
        WHEN report_category = 'tenure_type' THEN rt.
           rt_description
        ELSE report_subcategory
    END AS report_subcategory,
    report_totalAdultsCount,
    report_proportionPerceivingSecurity,
    report_proportionWithLegalDocumentation
FROM
    report_data r
LEFT JOIN
    la_humansexestype hst ON CAST(hst.hst_id AS TEXT) = r.
       report_subcategory AND r.report_category = 'gender'
LEFT JOIN
    la_righttype rt ON CAST(rt.rt_id AS TEXT) = r.
       report_subcategory AND r.report_category = 'tenure_type';
```

In this SQL view:

1. **Data Selection**: Data is retrieved for a specified time frame and geographic area by invoking the `SDG_1_4_2_Report` function.
2. **Code Translation**: The `LEFT JOIN` operations map the report_subcategory field's codes (such as gender and tenure type) to their descriptive labels from the `la_humansexestype` and `LA_Righttype` codelists.
3. **Output Formatting**: The final output is a formatted table where coded categories are substituted with meaningful descriptions, enhancing the report's readability and relevance.

This approach not only automates the data transformation process necessary for SDG reporting but also ensures that the reports are consistent, reducing the need for manual adjustments. The use of views streamlines data presentation and supports the generation of comprehensive reports that are readily understandable by a broad audience, thereby facilitating effective decision-making and policy formulation.

5. Testing

During the testing phase, the focus of this chapter is to validate the process of generating the SDG 1.4.2 indicator report by simulating land administration system data that is aligned with LADM. To ensure that the system can handle various scenarios in a real data environment, the test dataset is carefully designed to include both valid and invalid inputs. By simulating real-world operations, the insertion of test data in sequence verifies the correctness and effectiveness of the constraints, triggers, and functions implemented in SQL. The testing covers all stages of data insertion and updates, ensuring the system accurately captures changes in land tenure and generates an SDG 1.4.2 report in compliance with the established standards.

5.1. Test Dataset Design

5.1.1. Logic of Data Design

In reality, land administration systems are dynamic, with land rights, population, and spatial units changing over time. This dynamic nature was taken into account during the database design process, and thus, the test dataset is designed to accurately simulate the temporal changes of these elements. The data design logic is based on multi-dimensional time series data, including birth and death of individuals, transfer of land rights, and changes in spatial units. By using beginLifespanVersion and endLifespanVersion to manage the lifecycle of each record, the test data can faithfully reflect the dynamic changes of these elements at different points in time. The specific design considerations are as follows:

Geographic Dimension

The geographic dimension is a crucial part of the data design. The dataset contains three levels of geographic data:

- **Base Land Parcel**: The lowest level is represented by suid, which denotes individual land parcels, simulating the smallest spatial unit in reality.
- **Aggregated Region**: The next level is formed by aggregating several base land parcels into regions, denoted by extid.
- **Complete Test Area**: The highest level consists of all regions aggregated together, represented by test_area with id = 1.

The hierarchical geographic relationships are stored in the extaddress_suid_relation table, reflecting the connections between different geographic layers. This design simulates real-world scenarios where SDG indicator reports can be generated at various geographic scales, from national to local levels.

Temporal Dimension

The temporal dimension is the core of the data design, simulating the dynamic changes in land rights, population, and spatial units over time. The test data spans three years, with each year representing updates based on different changes to these elements. This design allows for a realistic replication of events and developments in a real system.

Year 1: System Initialization
The first year serves as the initialization phase, with population, land rights, and spatial unit data being set up. The design of the dataset reflects the demographic and spatial elements of the system, ensuring that the test data can accurately simulate real-world dynamics. The initial setup can be broken down as follows:

- **Party Data**: In total, the census records 22 individuals, of which 20 reside within the designated area and 2 are outside. As of January 1, 2000, 17 of these individuals are adults, while 3 are minors. This information is recorded in the extparty table, since the census is updated every ten years, the data in this table remain constant.. Of the 17 adults, 8 are recorded in the land administration system, with their information stored in the la_party table. These 8 individuals consist of 4 males and 4 females, which is also reflected in the la_party table.
- **Spatial Data**: The spatial dimension consists of 13 parcels, organized into two regions, with spatial information stored in the la_spatialunit table. Among the 8 adults with land ownership, 7 individuals each own one parcel, while 1 individual owns two parcels. The remaining 4 parcels are owned by the government, and this information is stored in the la_right and la_administrativesource tables.
- **Land Rights and Legal Documentation**: The land rights of the 8 adults and the 4 government-owned parcels are recorded in the la_right table. For each right, there is a corresponding legal document registered in the la_administrativesource table, ensuring that the legal basis for each land right is properly documented.
- **Secure Land Rights Questionnaire**: In addition, the extsecurelandrightsquestionnair table contains survey data for 10 individuals regarding their perception of land tenure security, reflecting data collected from the local population.

Year 2: Land Rights Transfer and Formation of a Cooperative
In the second year, changes in land rights and the formation of a cooperative are introduced to simulate more complex dynamics:

- **Land Rights Transfer**: Ownership of land parcel suid=30004 is transferred from pid=1004 to pid=10013. This change is recorded in multiple tables:
  - A new record for pid=10013 is added to the la_party table.
  - The la_right table reflects the transfer, where the original owner pid=10004 has their rights (identified by rid=20004) terminated, and a new right (identified by rid=20014) is created for the new owner pid=10013.
  - A new legal document for this transfer is added to the la_administrativesource table.

- **Formation of a Cooperative**: Three individuals form the farmerCooperation cooperative, with related information stored in the la_partygroup and la_partymember tables. This simulates group land management rights.

Year 3: Changes in Cooperative Membership and Land Leasing
In the third year, changes in cooperative membership and land leasing activities introduce further dynamics:

- **Changes in Cooperative Membership**: Membership in the cooperative changes, with some members leaving and new members joining. Additionally, the cooperative leases a parcel of land (suid=30013) from the government. This collective land leasing activity is recorded in the la_right table as a single record.
- **Individual Land Leasing**: In contrast to the cooperative, individuals (pid=10016, 10017, 10018, 10020) also lease the same parcel of land from the government, with each individual's lease recorded separately in the la_right table. This distinction highlights the difference between collective and individual leasing.
- **Legal Document Updates**: The la_administrativesource table is updated accordingly, recording both collective and individual land leases.
- **External Data Update**: Since secure land rights surveys are conducted every three to five years, the extsecurelandrightsquestionnair table is updated in the third year, adding records for three more individuals. According to the record of birthday, the number of minors decreases over time, from three in the first year, to two in the second year, and one in the third year.

By integrating both geographic and temporal dimensions, the test data effectively simulates the dynamic changes in land rights, population, and spatial units over time. This design ensures that the test data can accurately reflect real-world dynamics at different points in time, providing a robust foundation for subsequent functionality testing.

5.1.2. Data Insertion Order and Handling of Foreign Key Constraints

When inserting data into the database, the complex foreign key constraints require careful adherence to the dependency relationships between tables. For instance, when inserting data into the la_administrativesource table, several other related tables must have their data inserted first to satisfy the foreign key dependencies. Specifically, the following fields in the la_administrativesource table are constrained by foreign keys:

- suid references the la_spatialunit table.
- rid references the la_right table.
- pid references the la_party table.
- as_type references the la_administrativesourcetype table.

Thus, before inserting data into the la_administrativesource table, the corresponding records in la_spatialunit, la_right, la_party, and la_administrativesourcetype must be present in the database.

To ensure proper handling of foreign key constraints, the insertion of data follows this sequence: first, spatial data are inserted into the la_spatialunit table, followed by the insertion

of address and population data into the extaddress_suid_relation and extparty tables. Then, data are inserted into the LA_Party, la_right, and finally the la_administrativesource tables. The questionnaire data in the extsecurelandrightsquestionnaire table is inserted last. This insertion sequence ensures that all foreign key dependencies are properly addressed, preventing violations of constraint rules.

When updating existing data and inserting new records, especially during the second and third years of data operations, the following sequence must be observed:

- **Update existing data first**: Before inserting new records, any updates to existing records, particularly those involving the endLifespanVersion field, must be completed. This is crucial because many of the system's constraints depend on records where the endLifespanVersion field is still NULL. By updating the existing data first, these constraints will not interfere with subsequent operations.
- **Group insertion of new data**: When inserting new records, group them based on their relationships. For instance, when inserting new records into the la_right table, records with the same suid and r_type should be inserted together. This approach ensures data consistency and minimizes potential conflicts with foreign key constraints.

By carefully organizing the data insertion sequence, foreign key constraints are maintained, ensuring that all relationships within the system remain consistent.

5.2. Comparison of Automated and Manual Calculations

In this section, the results of the automated calculations performed by the database system over the three years are compared with the manually calculated outcomes. This comparison aims to validate the accuracy and consistency of the database implementation, ensuring that the complex constraints, functions, and relationships between tables work as intended to reflect the dynamic changes in land rights, population, and spatial units over time.

5.2.1. First Year

During the first year, the system generated the initial population and land rights data based on the inserted test dataset. The total adult population, the number of land parcels, and the corresponding land rights were calculated and stored. Manual calculations were performed to verify the following key results for the entire study area:

- **Total adult population**: The census dataset contained 22 records in total, out of which 20 individuals were located within the designated area. Among these 20 individuals, 17 were classified as adults, having reached the age of 18 by the reference date of January 1, 2000. The remaining 3 individuals were classified as minors, not yet 18 years old. This information was drawn from the extparty table.
- **Legally documented land rights**: Out of the 17 adults, 8 individuals—4 men and 4 women— all held ownership rights on land. These individuals had legal documentation for their land parcels, with all ownership rights recorded in the la_party and la_right tables. The 8 individuals only has one type of right, namely ownership. All this information, including the ownership structure, was confirmed in the la_right and la_administrativesource tables.

- **Perceived tenure security**: According to the questionnaire data stored in the extsecurelandrightsquestionnaire table, 10 individuals participated in the perceived tenure security survey. Of these 10 respondents, 8 (equally divided between 4 men and 4 women) reported feeling secure about their land rights. The remaining 2 individuals expressed uncertainty or insecurity regarding their tenure.

In summary, manual calculations yielded the following results: Of the total adult population in the designated areas, 23.53% of males and 23.53% of females had legally recognised land documents according to gender, and 47.06% had ownership on land according to tenure of type.When analysing the sense of security of tenure, 47.06% of the of the respondents said that they feel secure, with equal percentages of men and women.

The following are the results of the database calculations, generated by running the relevant compute functions. First, the computeProportionWithLegalDocumentation_number function is run to calculate the proportion of the adult population with legal land documentation.

```
SELECT *
FROM computeProportionWithLegalDocumentation_number(
    '2000-01-01 00:00:00',
    '2001-01-01 00:00:00',
    (SELECT geom FROM test_area WHERE id = 1)
);
```

By running this function for the complete test area in 2000, it provides a breakdown of the percentage of individuals who meet this criterion. The query results give insights into the extent of land documentation within the designated region for that time frame. The figure 5.1 shows the result of the computation, indicating the proportion of the adult population with legally documented land rights for the year 2000.

	category text	subcategory text	proportion double precision	total_adults_count integer	adults_with_documentation_count integer
1	total	[null]	0.47058823529411764	17	8
2	gender	9	0	17	0
3	gender	2	0.23529411764705882	17	4
4	gender	1	0.23529411764705882	17	4
5	tenure_type	11	0.47058823529411764	17	8

Figure 5.1.: Calculation of the percentage of legal land document holdings within the Complete Test Area in 2000

The next step is to use computeProportionPerceivingSecurity_number function calculating the proportion of adults who perceive their tenure rights as secure. The figure 5.2 show the results of the computation.

	category text	subcategory text	proportion double precision	total_adults_count integer	adults_perceiving_security_count integer
1	total	[null]	0.47058823529411764	17	8
2	gender	9	0	17	0
3	gender	2	0.23529411764705882	17	4
4	gender	1	0.23529411764705882	17	4
5	tenure_type	11	0.47058823529411764	17	8

Figure 5.2.: Calculation of the percentage of of adults who perceive their tenure rights as secure within the Complete Test Area in 2000

From these initial results, it can already be observed that the output matches the manually compiled statistics. The calculation of legally documented land rights and the perceived tenure security align with the manual count, confirming the accuracy of the functions.

Next, the SDG_1_4_2_Report_View_2000_all is executed to generate a comprehensive report, synthesizing the various metrics into a unified output. The report consolidates the data on legal documentation and tenure security, presenting it in a structured, readable format. By creating the view and comparing its output with the manually calculated results, the consistency and reliability of the automated database calculations can be further confirmed.

	report_begindate timestamp with time zone	report_enddate timestamp with time zone	report_ geome	report_category text	report_subcategory text	report_totaladultscount integer	report_proportionperceivingsecurity double precision	report_proportionwithlegaldocumentation double precision
1	2000-01-01 00:00:00+01	2001-01-01 00:00:00+01	01030000...	gender	Male	17	0.23529411764705882	0.23529411764705882
2	2000-01-01 00:00:00+01	2001-01-01 00:00:00+01	01030000...	tenure_type	ownership	17	0.47058823529411764	0.47058823529411764
3	2000-01-01 00:00:00+01	2001-01-01 00:00:00+01	01030000...	gender	Female	17	0.23529411764705882	0.23529411764705882
4	2000-01-01 00:00:00+01	2001-01-01 00:00:00+01	01030000...	gender	doesNotApply	17	0	0
5	2000-01-01 00:00:00+01	2001-01-01 00:00:00+01	01030000...	total	[null]	17	0.47058823529411764	0.47058823529411764

Figure 5.3.: Final Report View within the Complete Test Area in 2000

Once the view is created, the final report can be generated by querying the view and comparing the results with the manually calculated figures. These results are consistent with the manually verified data and confirm the validity and accuracy of the system's calculations.

5.2.2. Second Year

In the second year, significant changes occurred in both land rights and the composition of party groups, including the introduction of party groups (e.g., a farmer cooperation) and land ownership transfers. As with the first year, the database calculations were cross-referenced with manually compiled results for validation.

- **Total adult population**: The census dataset contained 22 records in total, out of which 20 individuals were located within the designated area. Among these 20 individuals, 17 were classified as adults, having reached the age of 18 by the reference date of January 1, 2000. The remaining 3 individuals were classified as minors, not yet 18 years old. This information was drawn from the extparty table.

- **Legally documented land rights**: Out of the 17 adults, 8 individuals—4 men and 4 women— all held ownership rights on land. These individuals had legal documentation for their land parcels, with all ownership rights recorded in the la_party and la_right tables. The 8 individuals only has one type of right, namely ownership. All this information, including the ownership structure, was confirmed in the la_right and la_administrativesource tables.
- **Perceived tenure security**: According to the questionnaire data stored in the extsecurelandrightsquestionnaire table, 10 individuals participated in the perceived tenure security survey. Of these 10 respondents, 8 (equally divided between 4 men and 4 women) reported feeling secure about their land rights. The remaining 2 individuals expressed uncertainty or insecurity regarding their tenure.

In summary, manual calculations yielded the following results: Of the total adult population in the designated areas, 23.53% of males and 23.53% of females had legally recognised land documents according to gender, and 47.06% had ownership on land according to tenure of type.When analysing the sense of security of tenure, 47.06% of the of the respondents said that they feel secure, with equal percentages of men and women.

- **Total adult population**: The census dataset still included 22 records, with 20 individuals residing in the designated area. By January 1, 2001, one person (extpid=110015, birthday is on 1982/2/17) reached adulthood, increasing the number of adults to 18. The remaining 2 individuals were minors, under 18.
- **Legally documented land rights**: Among the 18 adults, 9 individuals (4 men and 5 women) held ownership rights over land, representing 50% of the adult population. Specifically, men accounted for 22.22% and women for 27.78% of those with documented ownership rights. All land rights were classified as ownership.
- **Perceived tenure security**: Of the 10 respondents to the tenure security survey, 8 individuals (4 men and 4 women) felt secure about their land rights, comprising 44.44% of the total adult population. Both men and women equally represented 22.22% of those who reported feeling secure.

Summary: The second year's data reflects a slight increase in the adult population to 18, with 50% holding legal ownership rights and 44.44% reporting a sense of security in their land tenure. The results align closely with the first year's outcomes, maintaining a stable proportion of those feeling secure about their land rights.

Set the time range to '2001-01-01 00:00:00' to '2002-01-01 00:00:00', with the geographical area still selected as the entire test area. The results from the SDG_1_4_2_Report_View_2001_all are as follows:

	report_begindate timestamp with time zone	report_enddate timestamp with time zone	report_ geome	report_category text	report_subcategory text	report_totaladultscount integer	report_proportionperceivingsecurity double precision	report_proportionwithlegaldocumentation double precision
1	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	gender	Male	18	0.2222222222222222	0.2222222222222222
2	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	tenure_type	ownership	18	0.5	0.4444444444444444
3	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	gender	Female	18	0.2777777777777778	0.2222222222222222
4	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	gender	doesNotApply	18	0	0
5	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	tenure_type	lease	18	0	0
6	2001-01-01 00:00:00+01	2002-01-01 00:00:00+01	01030000...	total	[null]	18	0.5	0.4444444444444444

Figure 5.4.: Final Report View within the Complete Test Area in 2001

Also, match with manually calculated results.

5.2.3. Third Year

As above, the database calculations were cross-referenced with manually compiled results for validation.

- **Total adult population**: The total population did not change, but one additional individual (extpid=110022, born on December 3, 1983) reached adulthood, bringing the total number of adults to 19. This change slightly adjusted the demographics and affected the subsequent calculations regarding land rights and tenure security.
- **Legally documented land rights**: Out of the 19 adults, 12 individuals held legally documented land rights, representing 63.16% of the adult population. This group was composed of 4 men and 8 women, with percentages of 21.05% and 42.11%, respectively. When categorized by the type of rights, 9 individuals held ownership rights (47.37% of the total adult population), while 4 individuals had lease rights (21.05%).
- **Perceived tenure security**: The third year saw an addition of three new perceived tenure security records, of which two indicated a sense of security. This brought the total number of individuals perceiving their rights as secure to 10. However, only 9 of these individuals actually held legal rights to the land (extpid=110012 perceived security but did not possess any rights). Breaking it down by gender, 4 men and 5 women reported feeling secure, corresponding to 21.05% and 26.32% of the total adult population, respectively. Based on ownership, the percentage of individuals with secure tenure remains at 47.37%, reflecting the proportion of ownership rights holders who perceive their tenure as secure.

In the third year, the total adult population reached 19. Of these, 63.16% had legally documented land rights, with 21.05% being men and 42.11% being women. Ownership rights accounted for 47.37%, while lease rights made up 21.05%. Regarding tenure security, 52.63% of adults felt secure about their rights, with 21.05% of men and 26.32% of women expressing security, predominantly among those with ownership.

Set the time range to '2002-01-01 00:00:00' to '2003-01-01 00:00:00', with the geographical area still selected as the entire test area. The results from the SDG_1_4_2_Report_View_2002_all are as follows:

	report_begindate timestamp with time zone	report_enddate timestamp with time zone	report geome	report_category text	report_subcategory text	report_totaladultscount integer	report_proportionperceivingsecurity double precision	report_proportionwithlegaldocumentation double precision
1	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	gender	Male	19	0.21052631578947367	0.42105263157894735
2	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	tenure_type	ownership	19	0.47368421052631576	0.42105263157894735
3	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	gender	Female	19	0.2631578947368421	0.21052631578947367
4	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	gender	doesNotApply	19	0	0
5	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	tenure_type	lease	19	0	0.21052631578947367
6	2002-01-01 00:00:00+01	2003-01-01 00:00:00+01	01030000...	total	[null]	19	0.47368421052631576	0.631578947368421

Figure 5.5.: Final Report View within the Complete Test Area in 2002

Also match with manual calculated results.

SDG Report Visualization

In the third-year testing phase, SDG report maps were generated based on different attributes to showcase how various regions performed in land management indicators.

First, all parcels were divided into two regions based on their extaddressid attribute: Region 50001 is represented in red, and Region 50002 is represented in blue (shown in Figure 5.6a. These two regions correspond to different administrative or management units, displaying the specific SDG indicator performance for each area.

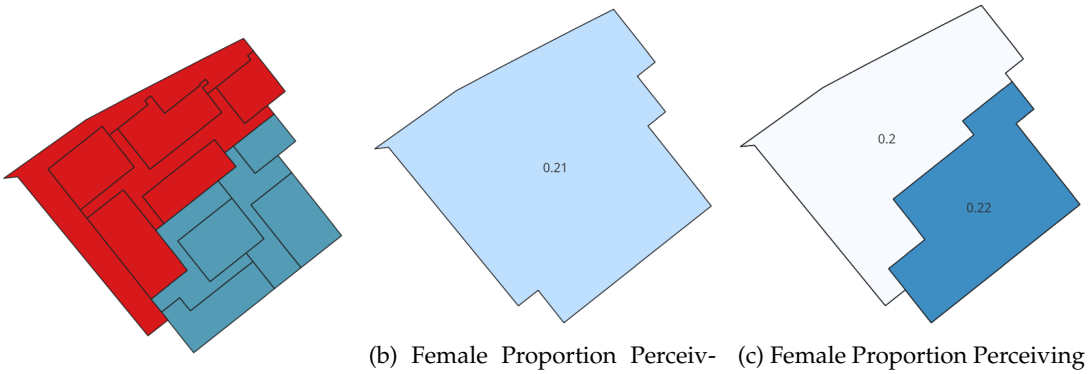
For these two regions, a total of 10 SDG report maps were generated for each, covering the following contents:

- The overall proportion perceiving security;
- Proportion perceiving security by gender (male/female);
- Proportion perceiving security by ownership type (ownership/lease);
- The overall proportion with legal documentation;
- Proportion with legal documentation by gender (male/female);
- Proportion with legal documentation by ownership type (ownership/lease).

These maps allow for a clear comparison of how Region 50001 and Region 50002 perform across these key SDG indicators (see Figure 5.7). The maps effectively illustrate the differences in performance across various attributes, providing valuable data to support land management policies.

In addition, an overall SDG report for the entire study area was generated, simulating a multi-level governance model. By combining the data from Regions 50001 and 50002, a higher-level regional report was produced, showing the aggregated performance across both areas (see Figure 5.6b, 5.6c). This multi-level reporting approach not only demonstrates the comparison between different layers but also lays the groundwork for future spatial analysis and data integration for additional parcels at the same level.

This multi-level reporting approach not only illustrates the comparison between different regions under the same topic but also lays the groundwork for future spatial and comparative analyses by incorporating more parcels at the same administrative level. With a richer set of spatial data, this analysis can significantly enhance monitoring of SDG indicators within regions and provide strong support for policy-making.



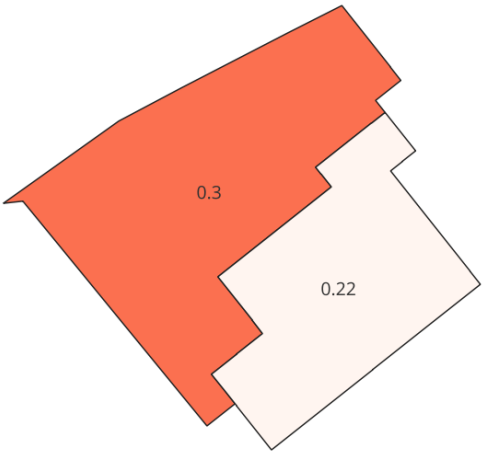
(a) Multi-Level Map

(b) Female Proportion Perceiving Security in First-level Map

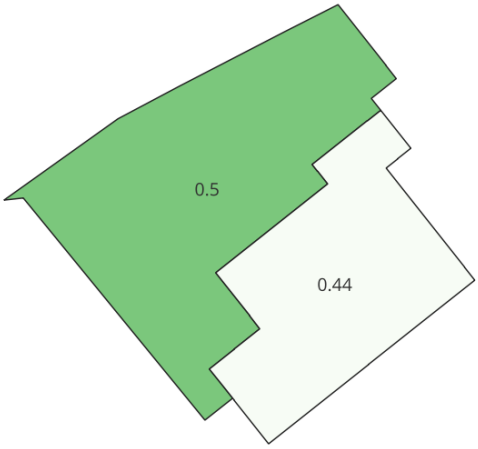
(c) Female Proportion Perceiving Security in second-level Map

Figure 5.6.: Multi-level Governance SDG Report Visualization for Combined and Sub-regions

(a) Overall Proportion Perceiving Security



(b) Female Proportion Perceiving Security



(c) Ownership Proportion Perceiving Security

(d) Overall Proportion with Legal Documentation

Figure 5.7.: Key SDG Report Maps for Regions 50001 and 50002

5.3. Validation of Constraints through Invalid Data Testing

This section validates the effectiveness of the system's constraint mechanisms by inserting invalid data into the database. The goal of this testing is to trigger various constraints and ensure that the system can effectively prevent erroneous or inconsistent data from being inserted. The tests are designed around different functions and constraints within the system, and the results demonstrate the robustness of these constraints in maintaining data integrity.

5.3.1. Testing check_version_lifespan_continuity

This test focuses on ensuring continuity in versioned records. Specifically, it validates that the beginLifespanVersion of a new record follows the endLifespanVersion of the previous record. Attempting to insert an invalid record, where the beginLifespanVersion is not continuous with the previous endLifespanVersion, correctly triggered the constraint and the insertion failed.

Invalid record example

First, a valid record was updated to have an endLifespanVersion, after which an attempt was made to insert a new record with a beginLifespanVersion ('2001-01-02 08:35:03') that did not match the corresponding endLifespanVersion ('2001-01-01 08:35:03'). Valid update to insert endLifespanVersion:

```
UPDATE LA_Right
SET
    endLifespanVersion = '2001-01-01 08:35:03',
    endRealWorldLifespanVersion = '2001-01-01 00:00:00'
WHERE
    rID = 20004;
```

Then, an invalid record is inserted where the beginLifespanVersion does not align with the previous record's endLifespanVersion, leading to an error:

```
INSERT INTO LA_Right (rID, r_type, suID, pID, share, shareCheck,
    timeSpec, beginLifespanVersion, endLifespanVersion,
   beginRealWorldLifespanVersion, endRealWorldLifespanVersion)
VALUES
(20014, 11, 30004, 10013, '(1,1)', TRUE, NULL, '2001-01-02
   08:35:03', NULL, '2001-01-01 00:00:00', NULL);
```

This operation will trigger the following error due to the mismatch in version continuity:

```
UPDATE LA_Right
SET
    endLifespanVersion = '2001-01-01 08:35:03',
    endRealWorldLifespanVersion = '2001-01-01 00:00:00'
WHERE
    rID = 20004;
UPDATE LA_Right
SET
    endLifespanVersion = '2001-01-01 08:35:03',
    endRealWorldLifespanVersion = '2001-01-01 00:00:00'
```

Data Output | Messages | Notifications

```
ERROR:  Error occurred in check_version_lifespan_continuity: Lifespan continuity error in table la_right for suid 30004: previous
endLifespanVersion 2001-01-01 08:35:03+01 does not match new beginLifespanVersion 2001-01-02 08:35:03+01
CONTEXT:  在RAISE的第45行的PL/pgSQL函数check_version_lifespan_continuity()

错误:  Error occurred in check_version_lifespan_continuity: Lifespan continuity error in table la_right for suid 30004: previous
endLifespanVersion 2001-01-01 08:35:03+01 does not match new beginLifespanVersion 2001-01-02 08:35:03+01
SQL state: P0001
```

Figure 5.8.: Error message caused by check_version_lifespan_continuity constraint

5.3.2. Testing check_fraction_validity

The purpose of this test was to validate the fraction constraints, ensuring that fractions have valid values for the numerator and denominator. The system successfully blocked invalid fractions (e.g., where the denominator was zero or the numerator was greater than the denominator), showing that the constraint was working as intended.

Denominator equals 0

The following SQL statement attempts to insert a record where the denominator of the share attribute is zero. This violates the constraint that the denominator must be greater than zero.

```
INSERT INTO LA_PartyMember (pmid, pgid, pid, share,
    beginLifespanVersion, endLifespanVersion,
    beginRealWorldLifespanVersion, endRealWorldLifespanVersion)
VALUES (80004, 70001, 10002, '(1,0)', '2001-10-01 10:24:38',
    NULL, '2001-10-01 00:00:00', NULL);
```

Query | Query History

```
INSERT INTO LA_PartyMember (pmid, pgid, pid, share, beginLifespanVersion, endLifespanVersion, beginReal
VALUES (80004, 70001, 10002, '(1,0)', '2001-10-01 10:24:38', NULL, '2001-10-01 00:00:00', NULL);
```

Data Output | Messages | Notifications

```
ERROR:  Error in record with pmid=80004: Denominator must be greater than zero, found 0
CONTEXT:  在RAISE的第19行的PL/pgSQL函数check_fraction_validity()
```

Figure 5.9.: Error message caused by denominator equal to 0

Numerator greater than denominator

This SQL statement attempts to insert a record where the numerator of the share is greater than the denominator. The constraint ensures that the numerator must be less than or equal to the denominator.

```
INSERT INTO LA_PartyMember (pmid, pgid, pid, share,
   beginLifespanVersion, endLifespanVersion,
   beginRealWorldLifespanVersion, endRealWorldLifespanVersion)
VALUES (80004, 70001, 10002, '(3,2)', '2001-10-01 10:24:38',
   NULL, '2001-10-01 00:00:00', NULL);
```

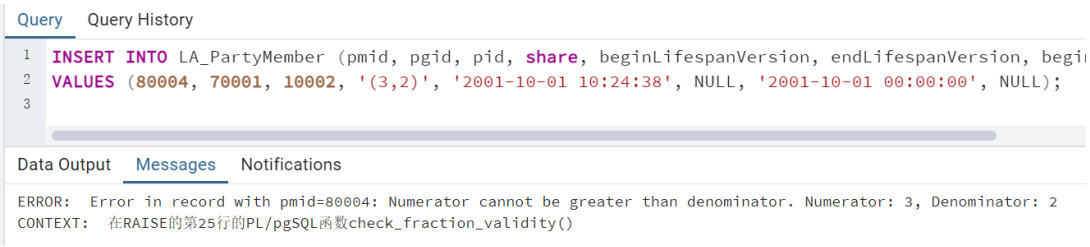


Figure 5.10.: Error messages caused by numerator greater than denominator

share value is NULL

This SQL statement attempts to insert a record where the share attribute is NULL. The constraint requires that the share value must be specified (i.e., not NULL).

```
INSERT INTO LA_PartyMember (pmid, pgid, pid, share,
   beginLifespanVersion, endLifespanVersion,
   beginRealWorldLifespanVersion, endRealWorldLifespanVersion)
        VALUES (80004, 70001, 10002, NULL, '2001-10-01 10:24:38'
           , NULL, '2001-10-01 00:00:00', NULL);
```

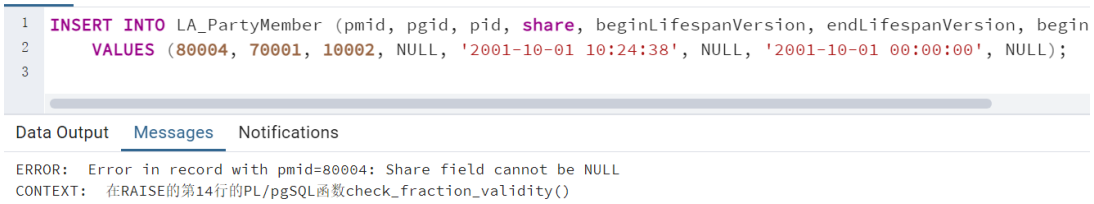


Figure 5.11.: Error message due to NULL share value

Negative numerator

This SQL statement attempts to insert a record where the numerator of the share attribute is negative. The constraint specifies that the numerator must be non-negative.

```
INSERT INTO LA_PartyMember (pmid, pgid, pid, share,
    beginLifespanVersion, endLifespanVersion,
    beginRealWorldLifespanVersion, endRealWorldLifespanVersion)
VALUES (80004, 70001, 10002, '(-3,2)', '2001-10-01 10:24:38',
    NULL, '2001-10-01 00:00:00', NULL);
```

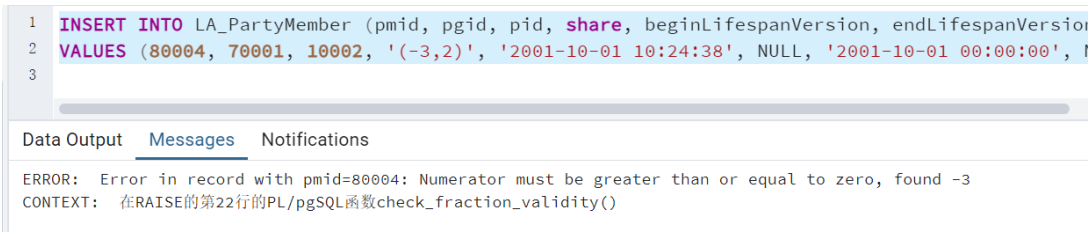


Figure 5.12.: Error message due to negative denominator

5.3.3. Testing check_minimum_group_members

The check_minimum_group_members constraint ensures that every group in the LA_PartyGroup must have at least two members. Attempting to create a group with fewer than two members will violate this constraint. Below is an example where an attempt is made to insert a group with only one member.

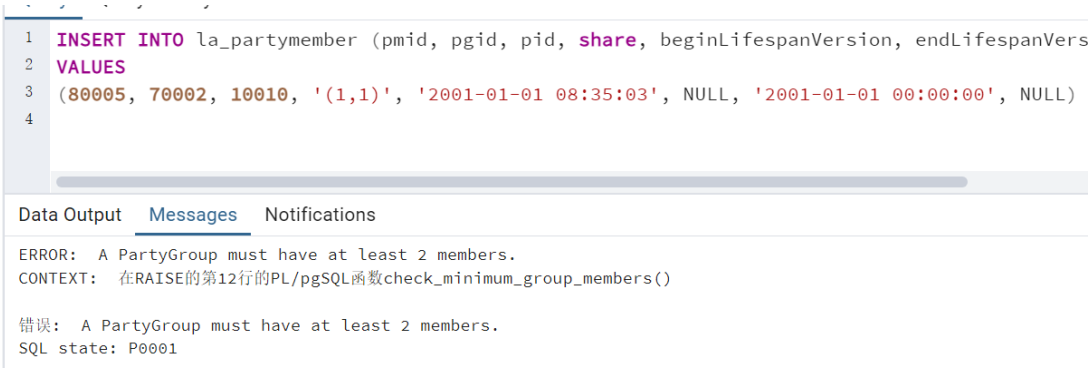


Figure 5.13.: Error message due to a group with only one member

5.3.4. Testing check_administrative_source_constraints

The check_administrative_source_constraints function ensures that records in the LA_AdministrativeSource table maintain consistency with related tables, specifically LA_Right. This function enforces that the timestamps and relationships between the administrative source data and the land rights records are correctly aligned, both for insertions and updates.

In the following example, we attempt to insert a record into the LA_AdministrativeSource table, but the data violates the constraints due to a mismatch between the lifeSpanStamp and the corresponding beginLifespanVersion in the LA_Right table.

Before testing the invalid case, a correct record should be inserted into the LA_Right table to set up the constraint checking:

```
UPDATE LA_Right
SET
    endLifespanVersion = '2001-01-01 08:35:03',
    endRealWorldLifespanVersion = '2001-01-01 00:00:00'
WHERE
    rID = 20004;
```

Then attempt to insert a record into LA_AdministrativeSource where the lifeSpanStamp does not match the beginLifespanVersion from the related LA_Right table, triggering a constraint violation:

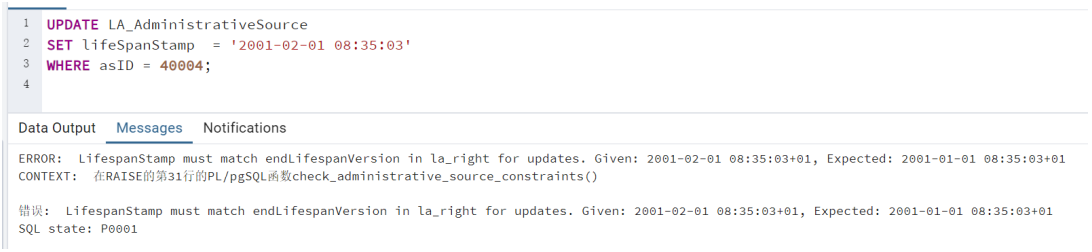


Figure 5.14.: Error message due to inconsistency between LA_AdministrativeSource information and LA_Right

6. Conclusion

6.1. Research overview

This research aimed to implement and evaluate a land administration system based on the ISO 19152 LADM, focusing on SDG indicator 1.4.2. The primary objective was to design a formalized land administration database architecture and dynamically assess its performance and accuracy to verify its potential for real-world applications. The key steps in this process included design, implementation, and system evaluation.

6.1.1. Design Phase

- **Conceptual model building**: Based on the LADM specification, a conceptual model was designed to specify the data structure of the core classes including LA_Party (Parties), LA_SpatialUnit (Spatial Units), and LA_RRR (Rights, Responsibilities and Restrictions). Attributes required to support the calculation of SDG 1.4.2 indicators, such as the legal authentication status of land tenure and the perceived security of land tenure, were specifically added to the model.
- **Model Conversion**: A detailed plan on how to convert the conceptual model into a physical database model, including data type selection, table structure design and relationship definition. Special attention was paid to the use of spatial data types to ensure that PostGIS can support spatial queries for land use. The architecture was designed with special consideration of data consistency, scalability and flexibility to ensure that the system can be adapted to the needs in different land administration scenarios.

6.1.2. Implementation Phase

- **Database Architecture Implementation**: The designed database schema was implemented in PostgreSQL, individual tables and their associations were constructed, and the necessary indexes were defined to optimise query performance. PostGIS extensions were used to handle the storage and querying of spatial data such as parcel boundaries.
- **Development of custom functions and triggers**: Several custom functions and database triggers were developed to ensure data consistency and implement complex business rules. These functions include automated calculation of land ownership percentages, validation of data integrity and implementation of business logic, such as management of the history of land rights changes. These functions and triggers are the core part of the system, ensuring operational accuracy and data reliability.

6.1.3. Evaluation Phase

To verify the system's effectiveness, a test dataset covering various land administration scenarios was generated. This dataset simulated land tenure transfers, population changes, and administrative boundary adjustments over different years. By simulating real-world land administration processes, the system's performance in handling complex business scenarios was tested. The evaluation focused on the accuracy, data consistency, and computational efficiency of the system when automatically calculating SDG 1.4.2 indicators, such as the coverage of legal documentation and perceived tenure security.

The system evaluation was not limited to normal data processing but also tested its robustness by inserting invalid data. The results showed that when invalid data or unreasonable land tenure changes occurred, the system triggered appropriate error handling mechanisms through pre-defined constraints. The system demonstrated high flexibility and scalability, quickly adapting to various data scenarios.

Throughout the evaluation, the study verified the system's accuracy and consistency by comparing automated calculation results with manual calculations. The dynamic three-year test demonstrated that the system could efficiently handle complex land tenure and spatial data while reliably generating reports aligned with SDG 1.4.2 requirements. The rigorous testing proved the system's effectiveness in managing large-scale land administration data, particularly in automating the calculation and monitoring of land tenure changes.

Overall, this study successfully demonstrated the potential application of the LADM-based database architecture in calculating SDG 1.4.2 indicators. Through formalized design and rigorous implementation and evaluation, the study validated the system's effectiveness and scalability in handling complex dynamic land data, providing a solid foundation for future land administration system development.

6.1.4. Answers to Subquestions

How can the conceptual model for SDG 1.4.2 be developed based on its metadata?

The conceptual model for SDG 1.4.2 was developed using the Four-Step Method. First, the SDG metadata was analyzed to identify key elements, like legal documentation and tenure security. These were then mapped to relevant LADM classes, ensuring alignment with the standard. The model was refined by adding necessary attributes and tested to confirm its accuracy in supporting SDG 1.4.2 indicator calculations, reducing the reliance on manual data collection and ensuring consistency.

How can the conceptual model for SDG 1.4.2 be effectively translated into a physical database implementation?

The translation from the conceptual model to a physical database was achieved using PostgreSQL and PostGIS. The physical model was implemented by defining tables that correspond to the core classes of the conceptual model. Specific attention was given to data types, table structures, and the relationships between spatial and non-spatial data to ensure the database was capable of handling the complex requirements of land administration. PostGIS extensions were employed to manage spatial queries and data, such as parcel

boundaries, while custom functions and triggers ensured data consistency and the correct execution of business logic, including the management of land rights history and ownership percentages. This implementation provided a solid foundation for the system's operational use.

What added value does this process bring to the overall monitoring and evaluation of SDG indicators?

This process significantly enhances the monitoring and evaluation of SDG indicators by reducing the reliance on manual data collection and calculations through the integration of cadastral systems. By formalizing and automating the process, it ensures greater accuracy and consistency in the calculation of key indicators, such as tenure security and legal documentation, eliminating errors that could arise from manual methods. Furthermore, the system allows for efficient reporting across different time periods, regions, and governance levels, providing policymakers with timely and reliable data to assess trends and policy impacts. Its scalability and adaptability also make it suitable for application across various legal frameworks, ensuring that it can be effectively implemented in diverse land administration contexts to support broader SDG monitoring efforts.

6.2. Contribution

The contribution of this study is mainly in three aspects. First, by adopting the ISO 19152 LADM, the study provides a standardised and systematic solution for the calculation of SDG indicators in land administrative systems, which promotes the standardisation and consistency of global land administration practices. Second, the study designs and implements an automated system capable of dynamically processing land rights changes, population dynamics, and administrative updates, which provides a new path for the efficient calculation and monitoring of SDG 1.4.2 indicators, and improves the accuracy and efficiency of land administration data processing. Finally, the study not only verifies the feasibility of automated calculations using simulated data, but also lays the foundation for future applications and extensions to other SDG indicators based on real data, which provides a strong reference basis for subsequent academic research and policy formulation.

6.3. Limitations

The limitations of this study can be summarized as follows. First, the test data used in this research were synthetic, which, although useful for validating the system's functionality and accuracy, may not fully capture the complexity of real-world scenarios. As a result, the system's performance in practical applications may differ from the simulation results. Second, the focus of this research was solely on SDG 1.4.2, demonstrating the applicability of LADM to this indicator, but other SDG indicators were not explored. Future studies should expand to other land-related SDG indicators to fully assess the broad applicability of the LADM framework. Additionally, due to data limitations, this study did not evaluate the system's performance across different regions and legal contexts, which restricts the generalizability of the findings across different countries. Therefore, future research should validate the

system with real-world cadastral data and consider regional differences in land administration systems. The system's ability to manage concurrent multi-user access was also not addressed in this study. Issues such as maintaining data consistency and integrity during simultaneous data insertions, updates, and deletions by multiple users were not considered. This may impact system performance and reliability in real-world environments, and future work should incorporate concurrency handling and optimization mechanisms to meet the complex demands of practical applications.

6.4. Future work

While this research demonstrated the applicability of LADM in calculating SDG indicators, several areas warrant further exploration.

6.4.1. Application to Other SDG Indicators

This research focused primarily on SDG 1.4.2, but the system could be extended to support the calculation and monitoring of other land-related SDG indicators, such as SDG 11 ("Sustainable Cities and Communities") or SDG 15 ("Life on Land"). By adapting and optimizing the current model, the LADM-based system could cater to a broader range of sustainable development data needs, creating a more comprehensive land management solution.

6.4.2. Integration of Real-World Data for Validation

The current study utilized simulated datasets to validate the system's functionality and effectiveness. Future work should involve the integration of real-world land administration data, particularly from different countries and regions. This will help to further validate the model's applicability in complex, dynamic real-world scenarios and identify potential areas for improvement. Collaborations with local governments or international organizations could enhance the system's evaluation under diverse legal and administrative contexts.

6.4.3. Enhancement of User Interface and Report Generation Capabilities

Although this research focused on the database architecture and data processing logic, future work could enhance the user interface to make the system more user-friendly for non-technical users. The report generation module could also be expanded to allow for diverse formats and content, tailored to the specific needs of different users. This would provide more sophisticated data analysis and visualization functionalities.

6.4.4. Further Performance Optimization

While this research improved the system's performance by optimizing database queries and index structures, future work could continue to explore performance optimization methods for larger datasets and more complex queries. Additionally, further research could assess the system's responsiveness and stability in high-concurrency, multi-user environments, ensuring its feasibility for large-scale applications.

By pursuing these areas of expansion and optimization, future work can further enhance the practicality and flexibility of the land administration system developed in this research, providing stronger technical support for global land management and the achievement of sustainable development goals.

A. Reproducibility self-assessment

A.1. Marks for each of the criteria

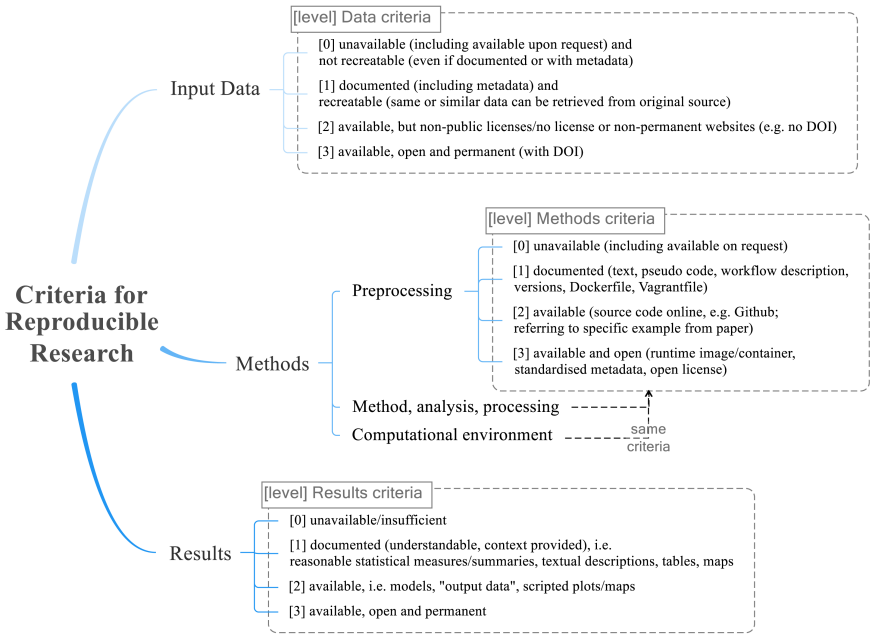


Figure A.1.: Reproducibility criteria to be assessed.

1. input data 3
2. preprocessing 3
3. methods 2
4. computational environment 3
5. results 3

A.2. Self-reflection

In my thesis, I aimed to ensure reproducibility through clear documentation and the use of accessible methods and tools. The data used were synthetic datasets, which are fully documented and available for reproduction. Since the data were not derived from real-world

sources, issues of confidentiality do not apply, and others can recreate similar datasets based on the provided metadata and assumptions.

All the methods, including database design, preprocessing, and SQL functions, are thoroughly documented. I used open-source tools such as PostgreSQL and PostGIS, ensuring others can replicate the work with similar setups. However, the use of real-world data was limited due to privacy and confidentiality concerns, which may restrict full replication in practical contexts. Nonetheless, the synthetic datasets and the reproducibility of the methods and outputs provide a strong foundation for future research to build upon.

B. Complete SQL code

This appendix provides all the code used in the research process, supporting the implementations and tests described in the main body of the thesis. The included code is primarily for building and testing the database system. The code samples are annotated to offer readers a clear understanding of the system's functionalities and structure, and to demonstrate practical data processing and analysis applications. Annotations are provided to enhance readability and usability, detailing the purpose and operation of each function and module.

```
create extension postgis;
```

Create a new data type by using `TYPE`

- Oid

```
CREATE TYPE Oid AS (
    localId VARCHAR,
    namespace VARCHAR
);
```

- Fraction

```
CREATE TYPE Fraction AS (
    numerator INTEGER,
    denominator INTEGER
);
```

B.1. Create Tables

B.1.1. Create codelists

LA_HumanSexesType

```
CREATE TABLE LA_HumanSexesType (
    hst_id INTEGER UNIQUE NOT NULL,
    hst_description TEXT NOT NULL UNIQUE
);
INSERT INTO LA_HumanSexesType (hst_id, hst_description ) VALUES
(0, 'unknown'),
(1, 'Male'),
(2, 'Female'),
(9, 'doesNotApply'),
(99, 'other');
```

LA_PartyType

```
CREATE TABLE LA_PartyType (
    pt_id SERIAL PRIMARY KEY,
    pt_description TEXT NOT NULL UNIQUE
);
INSERT INTO LA_PartyType (pt_description ) VALUES
('naturalPerson'),
('nonNaturalPerson'),
('baunit'),
('group');
```

LA_GroupPartyType

```
CREATE TABLE LA_GroupPartyType(
    gpt_id SERIAL PRIMARY KEY,
    gpt_description TEXT NOT NULL UNIQUE
);
INSERT INTO LA_GroupPartyType (gpt_description ) VALUES
('tribe'),
('association'),
('family'),
('baunitGroup'),
('municipality'),
('state'),
('farmerCooperation'),
('churchCommunity')
```

LA_RightType

```
CREATE TABLE LA_RightType (
    rt_id SERIAL PRIMARY KEY,
    rt_description TEXT NOT NULL UNIQUE
);
INSERT INTO LA_RightType (rt_description) VALUES
('agriActivity'),
('commonOwnership'),
('customaryType'),
('fireWood'),
('fishing'),
('grazing'),
('hunting'),
('informalOccupation'),
('lease'),
('occupation'),
('ownership'),
('ownershipAssumed'),
('superficies'),
('usufruct'),
('waterRights'),
('tenancy');
```

LA_AdministrativeSourceType

```
CREATE TABLE LA_AdministrativeSourceType(
    ast_id SERIAL PRIMARY KEY,
    ast_description TEXT NOT NULL UNIQUE
);

INSERT INTO LA_AdministrativeSourceType(ast_description) VALUES
('agriLease'),
('agriNotaryStatement'),
('deed'),
('mortgage'),
('title'),
('agriConsent');
```

ExtLandRightPerception

```
CREATE TABLE ExtLandRightPerception(
    elrp_id INTEGER PRIMARY KEY,
    elrp_description TEXT NOT NULL UNIQUE
);

INSERT INTO ExtLandRightPerception(elrp_id, elrp_description )
   VALUES
(0, 'insecure'),
(1, 'secure');
```

B.1.2. Create classes

Parent table for versionedobject

```
CREATE TABLE VersionedObject (
        beginLifespanVersion TIMESTAMPTZ,
        endLifespanVersion TIMESTAMPTZ,
        beginRealWorldLifespanVersion TIMESTAMPTZ,
        endRealWorldLifespanVersion TIMESTAMPTZ,

        CONSTRAINT chk_version_lifespan CHECK (
           endLifespanVersion IS NULL OR endLifespanVersion >
           beginLifespanVersion),
        CONSTRAINT chk_version_RealWorldLifespan CHECK (
           endRealWorldLifespanVersion IS NULL OR
           endRealWorldLifespanVersion >
           beginRealWorldLifespanVersion)
);
```

ExtParty

```
CREATE TABLE ExtParty (
        extPID INTEGER PRIMARY KEY,
        extAddressID INTEGER,
        birthday DATE,
        name TEXT,
        begindate TIMESTAMPTZ,
        enddate TIMESTAMPTZ
);
```

LA_Party

```
CREATE TABLE LA_Party (
    pID INTEGER PRIMARY KEY,
    extPID INTEGER REFERENCES ExtParty(extpID),
    name TEXT,
    humanSex INTEGER REFERENCES LA_HumanSexesType(hst_id),
    p_type INTEGER NOT NULL REFERENCES LA_PartyType(pt_id)
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

LA_PartyGroup

```
CREATE TABLE LA_PartyGroup (
    pgID INTEGER PRIMARY KEY,
    pid INTEGER NOT NULL REFERENCES LA_Party (pID),
    pg_type INTEGER NOT NULL REFERENCES LA_GroupPartyType (
       gpt_id)
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

LA_PartyMember

```
CREATE TABLE LA_PartyMember (
    pmID INTEGER PRIMARY KEY,
    pgID INTEGER NOT NULL REFERENCES LA_PartyGroup (pgID),
    pID INTEGER NOT NULL REFERENCES LA_Party (pID),
    share Fraction
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

LA_SpatialUnit

```
CREATE TABLE LA_SpatialUnit (
    suid SERIAL PRIMARY KEY,
    geom GEOMETRY,
    area DOUBLE PRECISION
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

LA_Right

```
CREATE TABLE LA_Right (
    rID INTEGER PRIMARY KEY,
    r_type INTEGER NOT NULL REFERENCES LA_RightType (rt_id),

    -- inherited from LA_RRR
    suID INTEGER REFERENCES la_spatialunit(suid),
    pID INTEGER REFERENCES la_party(pid),
    share fraction,
    shareCheck BOOLEAN,
    timeSpec TEXT
)
-- inherited from VersionedObject
INHERITS (VersionedObject);
```

LA_AdministrativeSource

```
CREATE TABLE LA_AdministrativeSource (
    asID INTEGER PRIMARY KEY,
    text TEXT,
    as_type INTEGER NOT NULL REFERENCES
       la_administrativesourcetype (ast_id),
    rid INTEGER REFERENCES la_right(rid),
    suid INTEGER REFERENCES la_spatialunit(suid),
    pid INTEGER REFERENCES la_party(pid),

    -- inherited from LA_Source
    acceptance TIMESTAMPTZ NOT NULL,
    lifeSpanStamp TIMESTAMPTZ NOT NULL
);
```

ExtSecureLandRightsQuestionnaire

```
CREATE TABLE ExtSecureLandRightsQuestionnaire (
    eslrq_id INTEGER PRIMARY KEY,
    selfperception INTEGER REFERENCES ExtLandRightPerception(
       elrp_id),
    name TEXT,
    extpid INTEGER,
    begindate TIMESTAMPTZ,
    enddate TIMESTAMPTZ
)
```

B.1.3. Create relations

extaddress_suid_relation

```
CREATE TABLE extaddress_suid_relation (
    extaddressid INTEGER NOT NULL,
    suid INTEGER NOT NULL,
    suid_geom geometry NOT NULL,
    PRIMARY KEY (extaddressid, suid),
    FOREIGN KEY (suid) REFERENCES la_spatialunit(suid)
);
```

B.2. Create Constraints

B.2.1. check_version_lifespan_continuity

```
CREATE OR REPLACE FUNCTION check_fraction_validity()
RETURNS TRIGGER AS $$
DECLARE
    pk_value TEXT;  -- Used to store the primary key value
    pk_name TEXT;   -- Used to store the primary key field name
       passed as a parameter
BEGIN
    -- Retrieve the primary key field name from the trigger's
       arguments
    pk_name := TG_ARGV[0];

    -- Dynamically retrieve the primary key value using the
       provided field name
    EXECUTE format('SELECT $1.%I FROM %I', pk_name,
       TG_TABLE_NAME) INTO pk_value USING NEW;

    -- Check if the share field is NOT NULL before proceeding
       with further checks
    IF NEW.share IS NOT NULL THEN
        -- Check if the denominator is greater than 0
        IF (NEW.share).denominator <= 0 THEN
            RAISE EXCEPTION 'Error in record with %=%:
               Denominator must be greater than zero, found %',
               pk_name, pk_value, (NEW.share).denominator;
        -- Check if the numerator is non-negative
        ELSIF (NEW.share).numerator < 0 THEN
            RAISE EXCEPTION 'Error in record with %=%: Numerator
                must be greater than or equal to zero, found %',
                pk_name, pk_value, (NEW.share).numerator;
        -- Check if the numerator is greater than the
           denominator
```

```
        ELSIF (NEW.share).numerator > (NEW.share).denominator
            THEN
             RAISE EXCEPTION 'Error in record with %=%: Numerator
                 cannot be greater than denominator. Numerator:
                %, Denominator: %',
                             pk_name, pk_value, (NEW.share).
                                numerator, (NEW.share).
                                denominator;
        END IF;
    END IF;

    -- Return the new or updated record if all checks pass
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Create triggers for the LA_Right

```
CREATE TRIGGER trg_check_version_lifespan_right
BEFORE INSERT OR UPDATE ON LA_Right
FOR EACH ROW
EXECUTE FUNCTION check_version_lifespan_continuity();
```

B.2.2. check_fraction_validity

```
CREATE OR REPLACE FUNCTION check_fraction_validity()
RETURNS TRIGGER AS $$
DECLARE
    pk_value TEXT;  -- Used to store the primary key value
    pk_name TEXT;   -- Used to store the primary key field name
        passed as a parameter
BEGIN
    -- Retrieve the primary key field name from the trigger's
        arguments
    pk_name := TG_ARGV[0];

    -- Dynamically retrieve the primary key value using the
        provided field name
    EXECUTE format('SELECT $1.%I FROM %I', pk_name,
        TG_TABLE_NAME) INTO pk_value USING NEW;

    -- Check if the share field is NOT NULL before proceeding
        with further checks
    IF NEW.share IS NOT NULL THEN
        -- Check if the denominator is greater than 0
        IF (NEW.share).denominator <= 0 THEN
```

```
            RAISE EXCEPTION 'Error in record with %=%:
               Denominator must be greater than zero, found %',
               pk_name, pk_value, (NEW.share).denominator;
        -- Check if the numerator is non-negative
        ELSIF (NEW.share).numerator < 0 THEN
            RAISE EXCEPTION 'Error in record with %=%: Numerator
                must be greater than or equal to zero, found %',
                pk_name, pk_value, (NEW.share).numerator;
        -- Check if the numerator is greater than the
           denominator
        ELSIF (NEW.share).numerator > (NEW.share).denominator
           THEN
            RAISE EXCEPTION 'Error in record with %=%: Numerator
                cannot be greater than denominator. Numerator:
               %, Denominator: %',
                           pk_name, pk_value, (NEW.share).
                              numerator, (NEW.share).
                              denominator;
        END IF;
    END IF;

    -- Return the new or updated record if all checks pass
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Create trigger for LA_Right

```
CREATE TRIGGER trg_check_fraction_validity_right
BEFORE INSERT OR UPDATE ON LA_Right
FOR EACH ROW
EXECUTE FUNCTION check_fraction_validity('rid');
```

Create trigger for LA_PartyMember

```
CREATE TRIGGER trg_check_fraction_validity_party_member
BEFORE INSERT OR UPDATE ON LA_PartyMember
FOR EACH ROW
EXECUTE FUNCTION check_fraction_validity('pmid');
```

B.2.3. check_share_sum

for LA_PartyMember

```
CREATE OR REPLACE FUNCTION check_share_sum_partymember() RETURNS
    VOID AS $$
DECLARE
    total_numerator INTEGER := 0;
    total_denominator INTEGER := 1;
    current_fraction RECORD;
BEGIN
    -- Query the sum of the shares under each pgID, making sure
       that endLifespanVersion is NULL.
    FOR current_fraction IN
        SELECT COALESCE((p.share).numerator, (t.share).numerator
            ) AS numerator,
                COALESCE((p.share).denominator, (t.share).
                    denominator) AS denominator
        FROM la_partymember p
        FULL JOIN temp_partymember t ON p.pgid = t.pgid
        WHERE (p.endLifespanVersion IS NULL)
    LOOP
        -- Debugging outputs the current numerator and
           denominator values
        RAISE NOTICE 'Current numerator: %, Current denominator:
            %', current_fraction.numerator, current_fraction.
           denominator;

        -- Update the logic of total_numerator and
           total_denominator.
        total_numerator := total_numerator * current_fraction.
           denominator + current_fraction.numerator *
           total_denominator;
        total_denominator := total_denominator *
           current_fraction.denominator;

        -- Debugging the output of the accumulated numerator and
            denominator values
        RAISE NOTICE 'Total numerator: %, Total denominator: %',
            total_numerator, total_denominator;
    END LOOP;

    -- Check if the sum of shares is 1
    IF total_numerator <> total_denominator THEN
        RAISE EXCEPTION 'Share sum error: total share for pgID
           is not equal to 1';
    ELSE
        -- If the check passes, insert the temporary table data
           into the main table
        INSERT INTO la_partymember (pmid, pgid, pid, share,
           beginLifespanVersion, endLifespanVersion,
           beginRealWorldLifespanVersion,
           endRealWorldLifespanVersion)
```

```
        SELECT pmid, pgid, pid, share, beginLifespanVersion,
           endLifespanVersion, beginRealWorldLifespanVersion,
           endRealWorldLifespanVersion
        FROM temp_partymember;

        RAISE NOTICE 'Data successfully inserted into
           la_partymember from temp_partymember';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

for LA_Right

```
CREATE OR REPLACE FUNCTION check_share_sum_laright() RETURNS
   VOID AS $$
DECLARE
    total_numerator INTEGER := 0;
    total_denominator INTEGER := 1;
    current_fraction RECORD;
BEGIN
    -- Query the sum of shares for each suID and r_type
       combination, making sure that endLifespanVersion is NULL.
    FOR current_fraction IN
        SELECT COALESCE((r.share).numerator, (t.share).numerator
           ) AS numerator,
               COALESCE((r.share).denominator, (t.share).
                  denominator) AS denominator
        FROM la_right r
        RIGHT JOIN temp_laright t ON r.suid = t.suid AND r.
           r_type = t.r_type
        WHERE (r.endLifespanVersion IS NULL and t.
           endLifespanVersion IS NULL)
    LOOP
        -- Debugging outputs the current numerator and
           denominator values
        RAISE NOTICE 'Current numerator: %, Current denominator:
            %', current_fraction.numerator, current_fraction.
           denominator;

        total_numerator := total_numerator * current_fraction.
           denominator + current_fraction.numerator *
           total_denominator;
        total_denominator := total_denominator *
           current_fraction.denominator;

        -- Debugging the output of the accumulated numerator and
            denominator values
```

```
        RAISE NOTICE 'Total numerator: %, Total denominator: %',
            total_numerator, total_denominator;
    END LOOP;

    -- Check if the sum of shares is 1
    IF total_numerator <> total_denominator THEN
        RAISE EXCEPTION 'Share sum error: total share for suID
            and r_type is not equal to 1';
    ELSE
        -- If the check passes, insert the temporary table data
            into the main table
        INSERT INTO la_right (rID, suID, r_type, pID, share,
            beginLifespanVersion, endLifespanVersion,
            beginRealWorldLifespanVersion,
            endRealWorldLifespanVersion)
        SELECT rID, suID, r_type, pID, share,
            beginLifespanVersion, endLifespanVersion,
            beginRealWorldLifespanVersion,
            endRealWorldLifespanVersion
        FROM temp_laright;

        RAISE NOTICE 'Data successfully inserted into la_right
            from temp_laright';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

B.2.4. check_minimum_group_members()

```
CREATE OR REPLACE FUNCTION check_minimum_group_members()
RETURNS TRIGGER AS $$
DECLARE
    member_count INTEGER;
BEGIN
    -- Count the number of members in the current PartyGroup
    SELECT COUNT(*) INTO member_count
    FROM LA_PartyMember
    WHERE pgID = NEW.pgID;

    -- Ensure that there are at least two members
    IF member_count < 2 THEN
        RAISE EXCEPTION 'A PartyGroup must have at least 2
            members.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Create trigger for LA_PartyMember

```
CREATE TRIGGER trg_check_minimum_group_members
AFTER INSERT OR UPDATE ON LA_PartyMember
FOR EACH ROW
EXECUTE FUNCTION check_minimum_group_members();
```

B.2.5. check_administrative_source_constraints

```
CREATE OR REPLACE FUNCTION
   check_administrative_source_constraints()
RETURNS TRIGGER AS $$
DECLARE
    existing_record RECORD;
    corresponding_right RECORD;
BEGIN
    -- Find out if the same asid exists (i.e. if it is an update
        operation)
    SELECT * INTO existing_record
    FROM LA_AdministrativeSource
    WHERE asid = NEW.asid;

    -- Find the corresponding la_right record
    SELECT * INTO corresponding_right
    FROM LA_Right
    WHERE rID = NEW.rID;

    -- Throws an error if the corresponding la_right record does
        not exist
    IF corresponding_right IS NULL THEN
        RAISE EXCEPTION 'Corresponding la_right record not found
            for rID %', NEW.rID;
    END IF;

    -- Check that acceptance is consistent with
       beginRealWorldLifespanVersion in the la_right table.
    IF NEW.acceptance IS DISTINCT FROM corresponding_right.
       beginrealworldlifespanversion THEN
        RAISE EXCEPTION 'Acceptance date must match
           beginRealWorldLifespanVersion in la_right. Given: %,
           Expected: %',
                        NEW.acceptance, corresponding_right.
                           beginrealworldlifespanversion;
    END IF;

    -- If the asid already exists, the update operation is
       performed
    IF existing_record IS NOT NULL THEN
```

```
        -- Ensure that lifeSpanStamp is the same as the
           endLifespanVersion of la_right at update time.
        IF NEW.lifespanstamp IS DISTINCT FROM
           corresponding_right.endlifespanversion THEN
            RAISE EXCEPTION 'LifespanStamp must match
               endLifespanVersion in la_right for updates. Given
               : %, Expected: %',
                           NEW.lifespanstamp,
                              corresponding_right.
                              endlifespanversion;
        END IF;
        -- Returns the updated record
        RETURN NEW;

    -- If the asid doesn't exist (i.e. it's an insertion
       operation), check if lifeSpanStamp is the same as
       beginLifespanVersion of la_right
    ELSE
        IF NEW.lifespanstamp IS DISTINCT FROM
           corresponding_right.beginlifespanversion THEN
            RAISE EXCEPTION 'LifespanStamp must match
               beginLifespanVersion in la_right for inserts.
               Given: %, Expected: %',
                           NEW.lifespanstamp,
                              corresponding_right.
                              beginlifespanversion;
        END IF;
        -- Returns the inserted new record
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Create trigger for LA_administrativesource

```
CREATE TRIGGER trg_check_administrative_source_constraints
BEFORE INSERT OR UPDATE ON la_administrativesource
FOR EACH ROW
EXECUTE FUNCTION check_administrative_source_constraints();
```

B.3. Create Functions

B.3.1. countAdult

```
-- only for extparty(census dataset)
CREATE OR REPLACE FUNCTION countAdults(CA_begindate DATE,
   CA_area GEOMETRY)
RETURNS INTEGER AS $$
DECLARE
    adultCount INTEGER := 0;
BEGIN
    SELECT COUNT(*) INTO adultCount
    FROM (
        SELECT DISTINCT ON (e.extpid) e.extpid
        FROM extparty e
        JOIN extaddress_suid_relation esr ON e.extaddressid =
           esr.extaddressid
        -- Ensure that the address is within the given
           geographical area
        WHERE ST_Contains(CA_area, esr.suid_geom)
        -- Calculation of the age of the adult, based on the
           date of commencement
        AND EXTRACT(YEAR FROM age(CA_begindate, e.birthday)) >=
           18
        -- Timeframe for checking census records
        AND (e.enddate IS NULL OR e.enddate >= CA_begindate)
        AND e.begindate <= CA_begindate
    ) AS unique_adults;

    RETURN adultCount;
END;
$$ LANGUAGE plpgsql;
```

B.3.2. computeProportionWithLegalDocumentation

output only proportion

```
CREATE OR REPLACE FUNCTION
   computeProportionWithLegalDocumentation(
    input_begin TIMESTAMPTZ,
    input_end TIMESTAMPTZ,
    input_area GEOMETRY
)
RETURNS TABLE (
    category TEXT,
    subcategory TEXT,
    proportion FLOAT
) AS $$
DECLARE
    totalAdultsCount INTEGER := 0;
    adultsWithDocumentationCount INTEGER := 0;
```

```
    gender INTEGER;
    tenureType INTEGER;
BEGIN
    -- Total number of adults
    totalAdultsCount := countAdults(input_begin::DATE,
       input_area);

    IF totalAdultsCount > 0 THEN
        -- Total number of adults with legal documents
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM(
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
            JOIN la_spatialunit s ON ads.suid = s.suid
            JOIN la_right r ON ads.rid = r.rid
            WHERE ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18 -- find the adult
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
 -- metadata "Reporting on the information contained in these
    land records ((i) namesof people holding rights, (ii) type
    of rights and (iii) location) is not difficult in principle
    if records are kept ina computerized format. "
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
               endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        RETURN QUERY SELECT 'total' AS category, NULL AS
           subcategory,
                            adultsWithDocumentationCount::FLOAT
                                / totalAdultsCount::FLOAT AS
                               proportion;
    ELSE
        RETURN QUERY SELECT 'total' AS category, NULL AS
           subcategory, 0::FLOAT AS proportion;
    END IF;

    -- divided by gender
    FOR gender IN SELECT DISTINCT humansex FROM la_party LOOP
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
```

```
            JOIN la_spatialunit s ON ads.suid = s.suid
            JOIN la_right r ON ads.rid = r.rid
            WHERE p.humansex = gender
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
               --make sure
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
               endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        IF totalAdultsCount > 0 THEN
            RETURN QUERY SELECT 'gender' AS category, gender::
               TEXT,
                                adultsWithDocumentationCount::
                                   FLOAT / totalAdultsCount::
                                   FLOAT AS proportion;
        ELSE
            RETURN QUERY SELECT 'gender' AS category, gender::
               TEXT, 0::FLOAT AS proportion;
        END IF;
    END LOOP;

    -- divided by tenure type
    FOR tenureType IN SELECT DISTINCT r_type FROM la_right LOOP
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
            JOIN la_right r ON ads.rid = r.rid
            JOIN la_spatialunit s ON r.suid = s.suid
            WHERE r.r_type= tenureType
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
               endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        IF totalAdultsCount > 0 THEN
```

```
            RETURN QUERY SELECT 'tenure_type' AS category,
               tenureType::TEXT,
                                adultsWithDocumentationCount::
                                   FLOAT / totalAdultsCount::
                                   FLOAT AS proportion;
        ELSE
            RETURN QUERY SELECT 'tenure_type' AS category,
               tenureType::TEXT, 0::FLOAT AS proportion;
        END IF;
    END LOOP;


END;
$$ LANGUAGE plpgsql;
```

output proportion and counted number

```
CREATE OR REPLACE FUNCTION
   computeProportionWithLegalDocumentation_number(
    input_begin TIMESTAMPTZ,
    input_end TIMESTAMPTZ,
    input_area GEOMETRY
)
RETURNS TABLE (
    category TEXT,
    subcategory TEXT,
    proportion FLOAT,
    total_adults_count INTEGER,
    adults_with_documentation_count INTEGER
) AS $$
DECLARE
    totalAdultsCount INTEGER := 0;
    adultsWithDocumentationCount INTEGER := 0;
    gender INTEGER;
    tenureType INTEGER;
BEGIN
    -- Total number of adults
    totalAdultsCount := countAdults(input_begin::DATE,
       input_area);

    IF totalAdultsCount > 0 THEN
        -- Total number of adults with legal documents
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM(
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
```

```
            JOIN la_spatialunit s ON ads.suid = s.suid
            JOIN la_right r ON ads.rid = r.rid
            WHERE ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
               endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        RETURN QUERY SELECT 'total' AS category, NULL AS
           subcategory,
                            adultsWithDocumentationCount::FLOAT
                                / totalAdultsCount::FLOAT AS
                               proportion,
                            totalAdultsCount,
                            adultsWithDocumentationCount;
    ELSE
        RETURN QUERY SELECT 'total' AS category, NULL AS
           subcategory, 0::FLOAT AS proportion,
                            totalAdultsCount,
                            0;
    END IF;

    -- divided by gender
    FOR gender IN SELECT DISTINCT humansex FROM la_party LOOP
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
            JOIN la_spatialunit s ON ads.suid = s.suid
            JOIN la_right r ON ads.rid = r.rid
            WHERE p.humansex = gender
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
               endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        IF totalAdultsCount > 0 THEN
```

```
            RETURN QUERY SELECT 'gender' AS category, gender::
                TEXT,
                                adultsWithDocumentationCount::
                                    FLOAT / totalAdultsCount::
                                    FLOAT AS proportion,
                                totalAdultsCount,
                                adultsWithDocumentationCount;
        ELSE
            RETURN QUERY SELECT 'gender' AS category, gender::
                TEXT, 0::FLOAT AS proportion,
                                totalAdultsCount,
                                0;
        END IF;
    END LOOP;

    -- divided by tenure type
    FOR tenureType IN SELECT DISTINCT r_type FROM la_right LOOP
        SELECT COUNT(*) INTO adultsWithDocumentationCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN la_administrativesource ads ON p.pid = ads.pid
            JOIN extparty ep ON p.extpid = ep.extpid
            JOIN la_right r ON ads.rid = r.rid
            JOIN la_spatialunit s ON r.suid = s.suid
            WHERE r.r_type = tenureType
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND (ads.pid IS NOT NULL AND ads.as_type IS NOT NULL
                AND ads.suid IS NOT NULL)
            AND (r.beginLifespanVersion <= input_end)
            AND (r.endLifespanVersion IS NULL OR r.
                endLifespanVersion >= input_begin)
        ) AS unique_adultsWithDocumentationCount;

        IF totalAdultsCount > 0 THEN
            RETURN QUERY SELECT 'tenure_type' AS category,
                tenureType::TEXT,
                                adultsWithDocumentationCount::
                                    FLOAT / totalAdultsCount::
                                    FLOAT AS proportion,
                                totalAdultsCount,
                                adultsWithDocumentationCount;
        ELSE
            RETURN QUERY SELECT 'tenure_type' AS category,
                tenureType::TEXT, 0::FLOAT AS proportion,
                                totalAdultsCount,
                                0;
```

```
        END IF;
    END LOOP;

END;
$$ LANGUAGE plpgsql;
```

B.3.3. computeProportionPerceivingSecurity

output only proportion

```
CREATE OR REPLACE FUNCTION computeProportionPerceivingSecurity(
    input_begin TIMESTAMPTZ,
    input_end TIMESTAMPTZ,
    input_area GEOMETRY
)
RETURNS TABLE (
    category TEXT,
    subcategory TEXT,
    proportion FLOAT
) AS $$
DECLARE
    totalAdultsCount INTEGER := 0;
    adultsPerceivingSecurityCount INTEGER := 0;
    gender INTEGER;
    tenureType INTEGER;
BEGIN
    -- Total number of adults
    totalAdultsCount := countAdults(input_begin::DATE,
       input_area);

    IF totalAdultsCount > 0 THEN
        -- Total Number of adults perceiving security
        SELECT COUNT(*) INTO adultsPerceivingSecurityCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN extsecurelandrightsquestionnaire slr ON p.
               extpid = slr.extpid
            JOIN la_right r ON p.pid = r.pid
            JOIN la_spatialunit s ON r.suid = s.suid
            JOIN extparty ep ON p.extpid = ep.extpid
            WHERE ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND slr.selfperception = 1
            AND (slr.begindate <= input_end::DATE)
```

```
            AND (slr.enddate IS NULL OR slr.enddate >=
                input_begin::DATE)
        ) AS unique_adultsPerceivingSecurityCount;

        RETURN QUERY SELECT 'total' AS category, NULL AS
            subcategory,
                            adultsPerceivingSecurityCount::
                                FLOAT / totalAdultsCount::FLOAT
                                AS proportion;
    ELSE
        RETURN QUERY SELECT 'total' AS category, NULL AS
            subcategory, 0::FLOAT AS proportion;
    END IF;

    -- divided by gender
    FOR gender IN (SELECT DISTINCT humansex FROM la_party) LOOP
        SELECT COUNT(*) INTO adultsPerceivingSecurityCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN extsecurelandrightsquestionnaire slr ON p.
                extpid = slr.extpid
            JOIN la_right r ON p.pid = r.pid
            JOIN la_spatialunit s ON r.suid = s.suid
            JOIN extparty ep ON p.extpid = ep.extpid
            WHERE p.humansex = gender
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND slr.selfperception = 1
            AND (slr.begindate <= input_end::DATE)
            AND (slr.enddate IS NULL OR slr.enddate >=
                input_begin::DATE)
        ) AS unique_adultsPerceivingSecurityCount;

        IF totalAdultsCount > 0 THEN
            RETURN QUERY SELECT 'gender' AS category, gender::
                TEXT,
                                adultsPerceivingSecurityCount::
                                    FLOAT / totalAdultsCount::
                                    FLOAT AS proportion;
        ELSE
            RETURN QUERY SELECT 'gender' AS category, gender::
                TEXT, 0::FLOAT AS proportion;
        END IF;
    END LOOP;

    -- divided by tenure type
```

```
    FOR tenureType IN (SELECT DISTINCT r_type FROM la_right)
       LOOP
        SELECT COUNT(*) INTO adultsPerceivingSecurityCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN extsecurelandrightsquestionnaire slr ON p.
               extpid = slr.extpid
            JOIN la_right r ON p.pid = r.pid
            JOIN la_spatialunit s ON r.suid = s.suid
            JOIN extparty ep ON p.extpid = ep.extpid
            WHERE r.r_type = tenureType
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND slr.selfperception = 1
            AND (slr.begindate <= input_end::DATE)
            AND (slr.enddate IS NULL OR slr.enddate >=
               input_begin::DATE)
        ) AS unique_adultsPerceivingSecurityCount;

        IF totalAdultsCount > 0 THEN
            RETURN QUERY SELECT 'tenure_type' AS category,
               tenureType::TEXT,
                                adultsPerceivingSecurityCount::
                                   FLOAT / totalAdultsCount::
                                   FLOAT AS proportion;
        ELSE
            RETURN QUERY SELECT 'tenure_type' AS category,
               tenureType::TEXT, 0::FLOAT AS proportion;
        END IF;
    END LOOP;

END;
$$ LANGUAGE plpgsql;
```

output proportion and counted number

```
CREATE OR REPLACE FUNCTION
   computeProportionPerceivingSecurity_number(
    input_begin TIMESTAMPTZ,
    input_end TIMESTAMPTZ,
    input_area GEOMETRY
)
RETURNS TABLE (
    category TEXT,
    subcategory TEXT,
```

```
    proportion FLOAT,
    total_adults_count INTEGER,
    adults_perceiving_security_count INTEGER
) AS $$
DECLARE
    totalAdultsCount INTEGER := 0;
    adultsPerceivingSecurityCount INTEGER := 0;
    gender INTEGER;
    tenureType INTEGER;
BEGIN
    -- Total number of adults
    totalAdultsCount := countAdults(input_begin::DATE,
       input_area);

    IF totalAdultsCount > 0 THEN
        -- Total Number of adults perceiving security
        SELECT COUNT(*) INTO adultsPerceivingSecurityCount
        FROM (
            SELECT DISTINCT ON (p.pid) p.pid
            FROM la_party p
            JOIN extsecurelandrightsquestionnaire slr ON p.
               extpid = slr.extpid
            JOIN la_right r ON p.pid = r.pid
            JOIN la_spatialunit s ON r.suid = s.suid
            JOIN extparty ep ON p.extpid = ep.extpid
            WHERE ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND slr.selfperception = 1
            AND (slr.begindate <= input_end::DATE)
            AND (slr.enddate IS NULL OR slr.enddate >=
               input_begin::DATE)
        ) AS unique_adultsPerceivingSecurityCount;

        RETURN QUERY SELECT
            'total' AS category,
            NULL AS subcategory,
            adultsPerceivingSecurityCount::FLOAT /
               totalAdultsCount::FLOAT AS proportion,
            totalAdultsCount,
            adultsPerceivingSecurityCount;
    ELSE
        RETURN QUERY SELECT
            'total' AS category,
            NULL AS subcategory,
            0::FLOAT AS proportion,
            totalAdultsCount,
            0;
    END IF;
```

```
-- divided by gender
FOR gender IN (SELECT DISTINCT humansex FROM la_party) LOOP
    SELECT COUNT(*) INTO adultsPerceivingSecurityCount
    FROM (
        SELECT DISTINCT ON (p.pid) p.pid
        FROM la_party p
        JOIN extsecurelandrightsquestionnaire slr ON p.
           extpid = slr.extpid
        JOIN la_right r ON p.pid = r.pid
        JOIN la_spatialunit s ON r.suid = s.suid
        JOIN extparty ep ON p.extpid = ep.extpid
        WHERE p.humansex = gender
        AND ST_Contains(input_area, s.geom)
        AND (ep.enddate IS NULL OR ep.enddate >= input_end)
        AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
            >= 18
        AND slr.selfperception = 1
        AND (slr.begindate <= input_end::DATE)
        AND (slr.enddate IS NULL OR slr.enddate >=
           input_begin::DATE)
    ) AS unique_adultsPerceivingSecurityCount;

    IF totalAdultsCount > 0 THEN
        RETURN QUERY SELECT
            'gender' AS category,
            gender::TEXT,
            adultsPerceivingSecurityCount::FLOAT /
               totalAdultsCount::FLOAT AS proportion,
            totalAdultsCount,
            adultsPerceivingSecurityCount;
    ELSE
        RETURN QUERY SELECT
            'gender' AS category,
            gender::TEXT,
            0::FLOAT AS proportion,
            totalAdultsCount,
            0;
    END IF;
END LOOP;

-- divided by tenure type
FOR tenureType IN (SELECT DISTINCT r_type FROM la_right)
   LOOP
    SELECT COUNT(*) INTO adultsPerceivingSecurityCount
    FROM (
        SELECT DISTINCT ON (p.pid) p.pid
        FROM la_party p
        JOIN extsecurelandrightsquestionnaire slr ON p.
           extpid = slr.extpid
```

```
            JOIN la_right r ON p.pid = r.pid
            JOIN la_spatialunit s ON r.suid = s.suid
            JOIN extparty ep ON p.extpid = ep.extpid
            WHERE r.r_type = tenureType
            AND ST_Contains(input_area, s.geom)
            AND (ep.enddate IS NULL OR ep.enddate >= input_end)
            AND EXTRACT(YEAR FROM age(input_begin, ep.birthday))
                >= 18
            AND slr.selfperception = 1
            AND (slr.begindate <= input_end::DATE)
            AND (slr.enddate IS NULL OR slr.enddate >=
               input_begin::DATE)
        ) AS unique_adultsPerceivingSecurityCount;

        IF totalAdultsCount > 0 THEN
            RETURN QUERY SELECT
                'tenure_type' AS category,
                tenureType::TEXT,
                adultsPerceivingSecurityCount::FLOAT /
                   totalAdultsCount::FLOAT AS proportion,
                totalAdultsCount,
                adultsPerceivingSecurityCount;
        ELSE
            RETURN QUERY SELECT
                'tenure_type' AS category,
                tenureType::TEXT,
                0::FLOAT AS proportion,
                totalAdultsCount,
                0;
        END IF;
    END LOOP;

END;
$$ LANGUAGE plpgsql;
```

B.3.4. SDG_1_4_2_Report

```
CREATE OR REPLACE FUNCTION SDG_1_4_2_Report(
    input_begin TIMESTAMPTZ,
    input_end TIMESTAMPTZ,
    input_area GEOMETRY
)
RETURNS TABLE (
    report_begindate TIMESTAMPTZ,
    report_enddate TIMESTAMPTZ,
    report_region geometry,
    report_category TEXT,
    report_subcategory TEXT,
```

```
    report_totalAdultsCount INTEGER,
    report_proportionPerceivingSecurity FLOAT,
    report_proportionWithLegalDocumentation FLOAT
) AS $$
DECLARE
    transformed_area GEOMETRY;
    totalAdultsCount INTEGER := 0;
BEGIN
    -- Transform input_area to the appropriate SRID
    transformed_area := ST_Transform(input_area, 28992);

    -- Compute the total number of adults once
    totalAdultsCount := countAdults(input_begin::DATE,
       transformed_area);

    -- Return the overall proportion
    RETURN QUERY
    SELECT
        input_begin AS report_begindate,
        input_end AS report_enddate,
        input_area AS report_region,
        'total' AS report_category,
        NULL AS report_subcategory,
        totalAdultsCount AS report_totalAdultsCount,
        (SELECT proportion FROM
           computeProportionPerceivingSecurity(input_begin,
           input_end, input_area) WHERE category = 'total') AS
           report_proportionPerceivingSecurity,
        (SELECT proportion FROM
           computeProportionWithLegalDocumentation(input_begin,
           input_end, input_area) WHERE category = 'total') AS
           report_proportionWithLegalDocumentation;

    -- Percentage by sex
    RETURN QUERY
    SELECT
        input_begin AS report_begindate,
        input_end AS report_enddate,
        input_area AS report_region,
        'gender' AS report_category,
        CAST(p.humansex AS TEXT) AS report_subcategory, --
           Convert humansex to TEXT type
        totalAdultsCount AS report_totalAdultsCount,
        (SELECT proportion FROM
           computeProportionPerceivingSecurity(input_begin,
           input_end, input_area) WHERE category = 'gender' AND
           subcategory = CAST(p.humansex AS TEXT)) AS
           report_proportionPerceivingSecurity, -- type
           conversion
```

```
        (SELECT proportion FROM
            computeProportionWithLegalDocumentation(input_begin,
            input_end, input_area) WHERE category = 'gender' AND
            subcategory = CAST(p.humansex AS TEXT)) AS
            report_proportionWithLegalDocumentation -- type
            conversion
    FROM
        la_party p
    GROUP BY
        p.humansex;

    -- Proportion by type of land tenure
    RETURN QUERY
    SELECT
        input_begin AS report_begindate,
        input_end AS report_enddate,
        input_area AS report_region,
        'tenure_type' AS report_category,
        CAST(r.r_type AS TEXT) AS report_subcategory, -- Convert
            r_type to TEXT type
        totalAdultsCount AS report_totalAdultsCount,
        (SELECT proportion FROM
            computeProportionPerceivingSecurity(input_begin,
            input_end, input_area) WHERE category = 'tenure_type'
             AND subcategory = CAST(r.r_type AS TEXT)) AS
            report_proportionPerceivingSecurity, -- type
            conversion
        (SELECT proportion FROM
            computeProportionWithLegalDocumentation(input_begin,
            input_end, input_area) WHERE category = 'tenure_type'
             AND subcategory = CAST(r.r_type AS TEXT)) AS
            report_proportionWithLegalDocumentation -- type
            conversion
    FROM
        la_right r
    GROUP BY
        r.r_type;
END;
$$ LANGUAGE plpgsql;
```

B.4. Test Data

The test data used in this research includes geospatial elements, which are extensive and complex, making them impractical to display directly in the document. All related test data, encompassing the initial data for the first year and the subsequent changes over the next two years, have been uploaded to GitHub. This data is crucial for validating the implementations and functionalities proposed in this thesis, especially in handling geospatial data and

executing complex spatial queries.

Interested readers can access and download these test data through the following link: GitHub - SDG-LADM Test-Data

Bibliography

Aditya, T., Sucaya, I. K. G. A., and Adi, F. N. (2021). Ladm-compliant field data collector for cadastral surveyors. *Land Use Policy*, 104:105356.

Agarwal, B. (2018). Gender equality, food security and the sustainable development goals. *Current opinion in environmental sustainability*, 34:26–32.

Alattas, A., van Oosterom, P., and Zlatanova, S. (2018a). Deriving the technical model for the indoor navigation prototype based on the integration of indoorgml and ladm conceptual model. In *7th International FIG Workshop on the Land Administration Domain Model*.

Alattas, A., van Oosterom, P., Zlatanova, S., Diakité, A. A., and Yan, J. (2018b). Developing a database for the ladm-indoorgml model.

Alattas, A., van Oosterom, P., Zlatanova, S., Diakité, A., and Yan, J. (2018c). Developing a database for the ladm-indoorgml model. In *6th International FIG 3D Cadastre Workshop*.

Augustinus, C. (2010). Social tenure domain model: what it can mean for the land industry and for the poor. In *XXIV FIG International Congress: facing the challenges, building the capacity*.

BECK, A., STOW, D., HILL, M., et al. (2021). Generic concepts to support country profiles, ladm implementation and indexing within formal land registers. *FIG e-Working Week*.

Bennett, R., Rajabifard, A., Williamson, I., and Wallace, J. (2012). On the need for national land administration infrastructures. *Land Use Policy*, 29(1):208–219.

Bennett, R. M., Unger, E.-M., Lemmen, C., and Dijkstra, P. (2021). Land administration maintenance: A review of the persistent problem and emerging fit-for-purpose solutions. *Land*, 10(5):509.

Bizoza, A. R. and Opio-Omoding, J. (2021). Assessing the impacts of land tenure regularization: Evidence from rwanda and ethiopia. *Land Use Policy*, 100:104904.

Bloomfield, G., Bucht, K., Martínez-Hernández, J. C., Ramírez-Soto, A. F., Sheseña-Hernández, I., Lucio-Palacio, C. R., and Ruelas Inzunza, E. (2018). Capacity building to advance the united nations sustainable development goals: An overview of tools and approaches related to sustainable land management. *Journal of sustainable forestry*, 37(2):157–177.

Carlson, M., Wells, J., and Jacobson, M. (2015). Balancing the relationship between protection and sustainable management in canada's boreal forest. *Conservation and Society*, 13(1):13–22.

Cf, O. (2015). Transforming our world: the 2030 agenda for sustainable development. *United Nations: New York, NY, USA*.

Chehrehbargh, F. J., Rajabifard, A., Atazadeh, B., Steudler, D., and Nugraha, B. W. (2024). Towards sustainable land governance: Extending the ladm to support global initiatives parameters - a case study in indonesia. In *12th International FIG Land Administration Domain Model & 3D Land Administration Workshop*.

Chen, J., Peng, S., Chen, H., Zhao, X., Ge, Y., and Li, Z. (2020). A comprehensive measurement of progress toward local sdgs with geospatial information: methodology and lessons learned. *ISPRS International Journal of Geo-Information*, 9(9):522.

Chen, M., Van Oosterom, P., Kalogianni, E., Dijkstra, P., and Lemmen, C. (2024). Bridging sustainable development goals and land administration: The role of the iso 19152 land administration domain model in sdg indicator formalization. *Land*, 13(4):491.

Chigbu, U. E. (2023). Connecting land tenure to land restoration. *Development in Practice*, 33(7):762–770.

Choudhury, P. R. and Behera, M. K. (2017). Using administrative data for monitoring and improving land policy and governance in india. In *Proceedings of the 10th International Conference on Theory and Practice of Electronic Governance*, pages 127–135.

Dale, P. and McLaughlin, J. (2000). *Land administration*. Oxford University Press.

Diaz-Sarachaga, J. M., Jato-Espino, D., and Castro-Fresno, D. (2018). Is the sustainable development goals (sdg) index an adequate framework to measure the progress of the 2030 agenda? *Sustainable Development*, 26(6):663–671.

Enemark, S., McLaren, R., and Lemmen, C. (2016). Fit-for-purpose land administration: guiding principles for country implementation. Ref. Doc. Version. Global Land Tool Network - UN-HABITAT, Nairobi, Kenya.

Enemark, S., Williamson, I., and Wallace, J. (2005). Building modern land administration systems in developed economies. *Journal of spatial science*, 50(2):51–68.

Feyertag, J., Childress, M., Langdown, I., Locke, A., and Nizalov, D. (2021). How does gender affect the perceived security of land and property rights? evidence from 33 countries. *Land Use Policy*, 104:105299.

for Standardization (ISO), I. O. (2024). Geographic information - land administration domain model (ladm) - part 1: Generic conceptual model. *International Organisation for Standardisation: Geneva, Switzerland*.

Gao, L. and Bryan, B. A. (2017). Finding pathways to national-scale land-sector sustainability. *Nature*, 544(7649):217–222.

Global Land Tool Network (2014). Social tenure domain model. https://stdm.gltn.net/. Accessed: 2024-9-7.

Global SDG Indicators (2024). Tier classification for global sdg indicators. Accessed: 6 Mar 2024.

Hák, T., Janoušková, S., and Moldan, B. (2016). Sustainable development goals: A need for relevant indicators. *Ecological indicators*, 60:565–573.

Hull, S., Kingwill, R., and Fokane, T. (2020). An introduction to land administration. *LandNNES: Cape Town, South Africa*.

Indrajit, A. (2019). 4d open spatial information infrastructure supporting participatory urban planning monitoring. Technical report, GISt Report. Accessed: 17 September 2024.

ISO (2019). Final report from stage 0 project on iso 19152 ladm. Iso/tc211/wg7n262, International Organization for Standardization (ISO), Geneva, Switzerland.

ISO, I. (2012). 19152: 2012.(2012). geographic information–land administration domain model (ladm). *International Organisation for Standardisation: Geneva, Switzerland*.

Kalfas, D., Kalogiannidis, S., Chatzitheodoridis, F., and Toska, E. (2023). Urbanization and land use planning for achieving the sustainable development goals (sdgs): A case study of greece. *Urban Science*, 7(2):43.

Kalogianni, E., Dimopoulou, E., Quak, W., Germann, M., Jenni, L., and Van Oosterom, P. (2017). Interlis language for modelling legal 3d spaces and physical 3d objects by including formalized implementable constraints and meaningful code lists. *ISPRS International Journal of Geo-Information*, 6(10):319.

Kalogianni, E., van Oosterom, P., Schmitz, M., Capua, R., Verbree, E., Dimopoulou, E., Gruler, H.-C., Stubkjær, E., Neudiens, I., Guarin, J. M., et al. (2023). Galileo high accuracy services: Support through iso 19162 ladm edition ii. In *FIG Working Week 2023: Protecting Our World, Conquering New Frontiers*.

Kara, A., Lemmen, C., van Oosterom, P., Kalogianni, E., Alattas, A., and Indrajit, A. (2024). Design of the new structure and capabilities of ladm edition ii including 3d aspects. *Land Use Policy*, 137:107003.

Kara, A., Rowland, A., van Oosterom, P., Stubkjaer, E., Cagdas, V., Folmer, E., Lemmen, C., Quak, W., and Meggiolaro, L. (2022). Formalisation of code lists and their values–the case of iso 19152 land administration domain model. In *10th Land Administration Domain Model Workshop*, pages 333–354. International Federation of Surveyors.

Katila, P., McDermott, C., Larson, A., Aggarwal, S., and Giessen, L. (2020). Forest tenure and the sustainable development goals–a critical view. *Forest Policy and Economics*, 120:102294.

Kim, R. E. (2023). Augment the sdg indicator framework. *Environmental Science & Policy*, 142:62–67.

Koeva, M., Stöcker, C., Crommelinck, S., Ho, S., Chipofya, M., Sahib, J., Bennett, R., Zevenbergen, J., Vosselman, G., Lemmen, C., et al. (2020). Innovative remote sensing methodologies for kenyan land tenure mapping. *Remote sensing*, 12(2):273.

Lawlor, K., Sills, E., Atmadja, S., Lin, L., and Songwathana, K. (2019). Sdg 1: No poverty–impacts of social protection, tenure security and building resilience on forests. *Sustainable development goals: their impacts on forests and people*, pages 17–47.

Lemmen, C. (2010). *The social tenure domain model: a pro-poor land tool: e-book.* International Federation of Surveyors (FIG).

Lisjak, J., Roić, M., Tomić, H., and Mastelić Ivić, S. (2021). Croatian ladm profile extension for state-owned agricultural land management. *Land*, 10(2):222.

Lyytimäki, J. (2019). Seeking sdg indicators. *Nature Sustainability*, 2(8):646–646.

Matsumoto, K., Hasegawa, T., Morita, K., and Fujimori, S. (2019). Synergy potential between climate change mitigation and forest conservation policies in the indonesian forest sector: implications for achieving multiple sustainable development objectives. *Sustainability Science*, 14:1657–1672.

Mbow, C. (2020). Use it sustainably or lose it! the land stakes in sdgs for sub-saharan africa. *Land*, 9(3):63.

Mengesha, A. K., Mansberger, R., Damyanovic, D., Agegnehu, S. K., and Stoeglehner, G. (2022). The contribution of land registration and certification program to implement sdgs: The case of the amhara region, ethiopia. *Land*, 12(1):93.

Molua, E. L. (2014). Land management for sustainable agriculture under climate change in the congo-basin countries of central africa. *ENVIRONMENT AND NATURAL RESOURCES RESEARCH*.

Morton, S., Pencheon, D., and Squires, N. (2017). Sustainable development goals (sdgs), and their implementation: A national global framework for health, development and equity needs a systems approach at every level. *British medical bulletin*, 124(1):81–90.

Mudau, N., Mwaniki, D., Tsoeleng, L., Mashalane, M., Beguy, D., and Ndugwa, R. (2020). Assessment of sdg indicator 11.3. 1 and urban growth trends of major and small cities in south africa. *Sustainability*, 12(17):7063.

Namubiru-Mwaura, E. (2014). Land tenure and gender: approaches and challenges for strengthening rural women's land rights. Accessed: 17 September 2024.

Sachs, J. D. (2012). From millennium development goals to sustainable development goals. *The lancet*, 379(9832):2206–2211.

Schmidt-Traub, G., Kroll, C., Teksoz, K., Durand-Delacre, D., and Sachs, J. D. (2017). National baselines for the sustainable development goals assessed in the sdg index and dashboards. *Nature geoscience*, 10(8):547–555.

Shahidinejad, J., Kalantari, M., and Rajabifard, A. (2024). Practical approaches to 3d cadastre implementation: Database schemas and exchange formats. In *12th International FIG Land Administration Domain Model & 3D Land Administration Workshop*.

Stabile, M. C., Guimarães, A. L., Silva, D. S., Ribeiro, V., Macedo, M. N., Coe, M. T., Pinto, E., Moutinho, P., and Alencar, A. (2020). Solving brazil's land use puzzle: Increasing production and slowing amazon deforestation. *Land use policy*, 91:104362.

Steudler, D., Rajabifard, A., and Williamson, I. P. (2004). Evaluation of land administration systems. *Land use policy*, 21(4):371–380.

Stubkjær, E., Paasch, J., Çağdaş, V., van Oosterom, P., Simmons, S., Paulsson, J., and Lemmen, C. (2018). International code list management: The case of land administration. In *7th International FIG Workshop on the Land Administration Domain Model 2018: in conjunction with Sixth Croatian Congress on Cadastre (VI. HKK)*.

Tan, E., Pattyn, V., Flores, C. C., and Crompvoets, J. (2021). A capacity assessment framework for the fit-for-purpose land administration systems: The use of unmanned aerial vehicle (uav) in rwanda and kenya. *Land Use Policy*, 102:105244.

Tirumala, R. D. and Tiwari, P. (2022). Importance of land in sdg policy instruments: A study of asean developing countries. *Land*, 11(2):218.

Tuholske, C., Gaughan, A. E., Sorichetta, A., de Sherbinin, A., Bucherie, A., Hultquist, C., Stevens, F., Kruczkiewicz, A., Huyck, C., and Yetman, G. (2021). Implications for tracking sdg indicator metrics with gridded population data. *Sustainability*, 13(13):7329.

Unger, E.-M., Bennett, R. M., Lemmen, C., and Zevenbergen, J. (2021). Ladm for sustainable development: An exploratory study on the application of domain-specific data models to support the sdgs. *Land Use Policy*, 108:105499.

Williamson, I., Enemark, S., and Wallace, J. (2006). Sustainability and land administration systems. *J. Wallace.–Department of Geomatics, Melbourne*, 271.

Williamson, I., Enemark, S., Wallace, J., and Rajabifard, A. (2010). *Land administration for sustainable development*. Citeseer.

Wunder, S., Kaphengst, T., and Frelih-Larsen, A. (2018). Implementing land degradation neutrality (sdg 15.3) at national level: general approach, indicator selection and experiences from germany. *International yearbook of soil law and policy 2017*, pages 191–219.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.