Secured and Modular Data Portal: Database System to Manage Broadly Classified and Large-Scale Data

PRACTICE PAPER

ATNAFU ABRHAM LENCHA
ADDISALEM BITEW MITIKU
ABEL TADESSE WOLDEMICHAEL

*Author affiliations can be found in the back matter of this article

ABSTRACT

Using various types of broadly classified and large-scale data, the Ethiopian Construction Design and Supervision Works Corporation (ECDSWC) provides professional services such as engineering studies and design. Storing, managing, and sharing datasets within workgroups and research teams was not an easy task before the corporation implemented the data portal. To resolve the issues related to data management, this study provides a secured and modular data portal by implementing REST API principles. The data portal is used to manage a wide variety of datasets, such as spatial data and their attributes, along with metadata and other serialized documents that support the business operation of the corporation. On the top of data management strategies, the Security in Depth (SiD) mechanism is developed by implementing the Unified Identity Authentication Service, which provides a multilayered security scheme. The applicability of the data portal is demonstrated using datasets obtained from ECDSWC. Further, the accuracy, integrity and privacy of the data are evaluated, and system performance is weighed through a prepared test case.

CORRESPONDING AUTHOR:
Atnafu Abrham Lencha
Surveying, Geospatial and Civil Informatics Center, Ethiopian Construction Design and Supervision Works Corporation, Addis Ababa, Ethiopia
dostyman71@gmail.com

KEYWORDS:
big data management; integrity; modules; performance optimization; REST API; security

TO CITE THIS ARTICLE:
Lencha, AA, Mitiku, AB and Woldemichael, AT. 2024. Secured and Modular Data Portal: Database System to Manage Broadly Classified and Large-Scale Data. *Data Science Journal*, 23: 20, pp. 1–17. DOI: https://doi.org/10.5334/dsj-2024-020

1. INTRODUCTION

In most African countries, the majority of data is recorded, published, and shared predominantly in hard copies. According to Amugongo et al. (2016), the impacts of digital data usage are found in a few African countries such as Kenya, Morocco, Tunisia, South Africa, Uganda, and Cameroon. On the other hand, in countries such as Ethiopia, very few groups of people can digitally store, share, and publish their data. The digital data used by those groups of people is unorganized, unmanaged, and difficult to share between different work groups. Further, with an increase in data volume generated from different projects and research works in these countries, there should be a robust and efficient data management system in place.

The Ethiopian Construction Design and Supervision Works Corporation (ECDSWC) is an Ethiopian government organization that provides professional services in multidisciplinary areas such as geology, meteorology, geodesy, hydrology, and soil and land evaluation. The organization demands an enormous volume of datasets to mobilize research and project work in the above-mentioned domains. Hence, it invests huge amount of money for research data collection. The collected data is large and broadly classified; as a result, it is difficult to manage using conventional methods. Moreover, the corporation must also manage documents such as standards, manuals, operational procedures, and project reports. Therefore, storing, managing, and sharing data within workgroups and research teams is not an easy task. Data anonymization, processing, integrity, and privacy are also the other challenges facing the corporation. Consequently, the corporation requires a data management system that is agile, modular, robust, and data-driven in order to store, manage, and share the datasets.

Prior evidence illustrates that data management plays a connecting role between data acquisition, data modelling, data visualization, and data analysis (Breunig et al. 2020). It also ensures continuous availability and reusability of data. Therefore, making digital data available for data users and preserving it accessible through online platform can benefit innovators as well as citizens to reuse their datasets (Amugongo et al. 2016). Prior research (Lnenicka et al. 2021) also argued that database technology can help data publishers to store all the relevant data in a managed fashion. Therefore, the structured, agile, and robust data management systems, depending on the purpose and discipline, can enable data users to acquire a high value of datasets (Lnenicka et al. 2021). Demand for data management, a growing number of data types, and the exponentially exploding volume of datasets (Wu et al. 2020) require scientists and researchers to find a scalable data integration and processing methods (Daquino et al. 2022; Pereira et al. 2022; Wu et al. 2020; Wu et al. 2022).

Data integration is the process used to provide a uniform view of datasets from the set of distributed, autonomous, and heterogeneous sources (Ji et al. 2019). Depending on the framework, the global schema can be necessary to produce a reconciled view of all available data from different sources (Ji et al. 2019). Various types of data integration techniques are available to deal with disparity, complexity, and heterogeneity of data sources. A well-known data integration techniques and approaches include ontological models (Wu et al. 2020); linked data platforms (Wu et al. 2022); data modeling, ingestion, and metadata extraction (Bauermeister et al. 2020); and federated query processing (Ji et al. 2019). Data integration is a complex process and comprises many sub-systems. This study emphasized the importance of data integration; therefore, the data portal needed to enfold data integration technique to exploit and visualize valuable knowledge and insights from heterogeneous and distributed datasets. However, for now the main goal of the study is to implement a scalable and robust database system.

To efficiently handle an exponentially exploding volume of data, this research prioritized the design and implementation of reliable and robust data storage as well as data-driven approaches to manage large and heterogeneous datasets. We reviewed different types of principles, technologies, and best practices that can help to store, manage, and share digital data. These technologies include, but are not limited to web services, remote method invocation (RMI), common object request broker architecture (CORBA), and distributed component object model (DCOM). When it comes to security or compatibility issues, technologies like RMI, CORBA, and DCOM can cause data breaches, making it challenging to share data through these platforms (Halili et al. 2018). Conversely, web services are platform independent and applicable in distributed and heterogeneous environments (Kumari et al. 2015). Therefore, they allow

multiple applications to be built via various programming languages and chain the module communication (Halili et al. 2018; Kumari et al. 2015). On top of that, the security scheme called Transport Layer Security (TLS) is used to secure web services (McGovern et al. 2003), supporting secure data sharing on the internet. TLS is the de facto standard to encrypt data between HTTP requests and responses (McGovern et al. 2003).

There are two main types of web services, namely, simple object access protocol (SOAP) and representational state transfer (REST) (Li et al. 2009; Mancini et al. 2022). In SOAP, web services have a unique web service description language (WSDL), which is used to document the contract between service provider and consumer. However, in REST, each service is represented in terms of its resources and each resource has a unique resource identifier (URI) through which services are accessed (Halili et al. 2018; Kumari et al. 2015). This makes REST preferable to SOAP. The other reason that REST outperforms SOAP is that REST is lightweight in nature, that is, REST can have message formats in JavaScript object notation (JSON) (Wu et al. 2022). As described in Wu et al. (2022), REST APIs can frequently change, are generally lightweight with few constraints, and are utilized as the building block for data communication in many web services. Consequently, multiple types of data repositories are currently using REST API (Kumari et al. 2015). On the other hand, SOAP relies on XML only, is verbose in nature, and increases the load on the server and network traffic (Halili et al. 2018). Parsing of SOAP messages can also intensify memory usage and computationally overload (Kumari et al. 2015).

To manage the large volumes of datasets and handle data transactions in the ECDSWC, this research work implements REST API principle encapsulated secured and modular data portal. The data portal is intended to manage raw data and other working documents together in a single platform. Within the data portal, raw data is organized, stored, and accessed in various thematic areas including geodesy, meteorology, hydrology, geology, and soil and land evaluation, and the objects of metadata are stored and can be edited when needed. Moreover, the data portal is planned to create a platform to store, organize, and access standards, manuals, operational procedures, and project reports. Therefore, the research teams and work groups in the corporation can store, retrieve, share, and manage relevant information, which supports business operations of the corporation and minimizes workload.

On top of data management strategies, data integrity and privacy, and system performance are also important. For the integrity, privacy, and user management, a unified identity authentication service (UIAS) is required, which can be achieved through configuration of the spring security framework and Microsoft's lightweight directory access protocol (LDAP). LDAP is an open, vendor-neutral, industry standard application protocol to access and maintain distributed directory information over an IP network (Wang et al. 2016). It is an x.500 standard with a hierarchical structure to store the data (Wu et al. 2014). The spring security framework is a highly customizable authentication and access control framework that provides powerful and customizable security features such as authentication and authorization (Islam et al. 2020). It is the de facto standard for securing Spring-based applications (Islam et al. 2020; Nguyen et al. 2019).

The other is database performance tuning, which improves overall system performance. The query processing overhead of database could be correlated with the whole system performance, which affects application design, code compilation, memory, and I/O strategy (Colley et al. 2017). In this regard, this study also examined various methods and principles for system and data storage performance tuning.

2. SECURED AND MODULAR DATA PORTAL

As discussed in the introduction section, various studies demonstrated the difficulties of storing, managing, and sharing large-sized, heterogeneous, and serialized binary data of different file formats. The exponential growth in the size of the dataset and the human-machine interaction also encountered integrity and privacy issues with the data, as well as system performance overhead. Developing a modular and integrated system to manage the datasets as well as their transactions at a research and engineering institute like ECDSWC is vital, considering a wide variety of issues as described above.

2.1. WHY IS THE DATA PORTAL REQUIRED FOR ECDSWC?

ECDSWC mobilizes multidisciplinary engineering and research work by using various types of broadly classified and large datasets. However, ECDSWC did not initially maintain an integrated data storage, sharing, and management system. This resulted in high work redundancy and significant budget spending while buying data repeatedly. On top of that, it was subject to data loss, breaches, security, and privacy issues.

To resolve these issues, this study created an opportunity to design and develop an integrated data management system that can store, manage, and share data, as well as to help in data reuse. Moreover, the system can be used for corresponding metadata clarification, data quality check, and input variable identification, which will resolve multiple problems that arise due to a lack of data management in the corporation (Mitiku et al. 2020). As an example, because of climate change in temporal and spatial variation, remote sensing and geo-observation technologies produce large volumes of data exponentially (Divac et al. 2009). As a result, the variety and velocity of recorded values of evolving spatial data have increased sharply over the last decade, resulting in uncertainty in all aspects of data management strategies (Züfle at al. 2020). In addition, the records of the spatial data attribute also increases the size of the data. Therefore, an agile, robust, and mature data management system needs to be in place. The other issues are privacy, integrity, and system performance. To improve system performance, database indexing, normalization, the REST API principle, thread pooling and file transfer via JSON file format are applied. To handle privacy, integrity, and user management, UIAS is achieved through the configuration of the spring security framework and the LDAP server. For access control, a role-based function has been implemented.

2.2. IMPLEMENTATION DETAILS

The design and implementation of a modular and secured data portal is needed to store, manage, and transfer data in the ECDSWC while ensuring data privacy and integrity. The significance of data integrity and privacy becomes even more serious when data is under attack (Pandey et al. 2020). Data integrity and privacy thus help to ensure the organization's brand. Integrity refers to maintaining and assuring the accuracy, and consistency of data, whereas privacy is about a proper handling, processing, storing, and usage of data (Pandey et al. 2020).

Therefore, this study develops a database system that can manage data and its transactions. It also implements a cross-platform authentication scheme known as UIAS. UIAS is the SiD mechanism that provides multilayered security measurements such as an implementation of TLS over HTTP protocol for web requests and creates a web-form that uses the 'POST' method. It also configures the secured socket layer (SSL) over LDAP server for identity validation and implements access privileges for data mapping and object binding. As shown in Figure 1, the enterprise system has many components and is designed into three distinct layers: presentation, service, and data access.

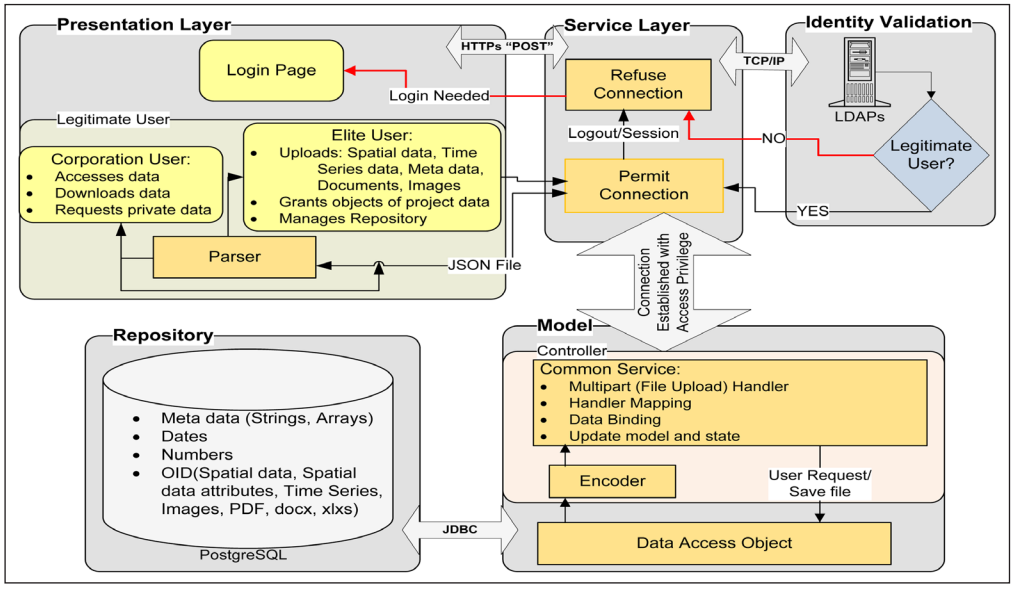


Figure 1 Modular and secured data portal architecture.

The Presentation layer implements user input and data access modules using Java Server Page (JSP), JavaScript (Js) and Cascading Style Sheet (CSS). To access, store and manage data in the repository, users need to log into the system. When the user is logged into the system, a connection is established with Read-Only and Read-Write privilege. The Read Only (R') privilege is granted to the 'Corporation User' to enable data access. The Read-Write (Wr') privilege is granted to the 'Elite User,' who can store metadata as well as binary data and manage data in the repository. This layer is also used to validate user inputs, file types and sizes. Moreover, it implements an asynchronous JavaScript and XML (AJAX) instance to handle and parse JSON files that are mapped from the service layer. Then the parsed file is displayed on web page in as a table, text message and binary data of different file formats. AJAX is implemented by using the `XMLHttpRequest()` JavaScript object, which is an abstract, built-in browser instance.

The service layer is used as a bridge to establish a secured connection between the presentation layer and model object, which uses HTTPs protocol and the 'POST' method. In addition to HTTPs configuration, the configuration of the spring security framework and the LDAP server is achieved, which carried out in Java XML class. This configuration creates a secured communication channel by implementing SSL over the LDAP server. In this configuration, an authentication manager `<sec:authentication-manager>` is a core interface in spring security framework that handles configuration procedures between the LDAP server and spring security framework. This layer establishes a secured channel and transmits the user identity credential from the presentation layer to the LDAP server via the TCP/IP protocol. If the user does not provide an identity credential, the LDAP establishes a default anonymous session; therefore, anonymous user can potentially access the resource (Halili et al. 2018; Wu et al. 2022). However, in this implementation, we prevented anonymous sessions to protect resources from unauthorized access.

The data access layer is the last layer of the architecture and is used to store ECDSWC data in the database. To deliver the data upon a user's demand, this layer connects to the model object via Java Database Connectivity (JDBC). The model object is a standalone and high-fidelity model of a real-world entity (Kumari et al. 2015). Both the data access instance and access privilege instance are implemented inside a model object based on the uniform interface of the REST API principle. The controller in the model object handles resource mapping, data binding, and file uploading between the data object and the presentation layer. Therefore, the resource mapping between the client and the data server is in a modular fashion. It is achieved by creating a layer of abstraction on the top of data access layer by defining resources that encapsulate entities of the data.

2.3. CONCEPTUAL DATA MODEL

The Conceptual Data Model (CDM) is a high-level representation of data that the organization uses. It explains the relationships between entities, rather than providing a detailed description of data about the business (West 2011). According to Batra et al. (1992) and Sherman (2015), CDMs focus on identifying entities, attributes, categories, and relationships between the entities but not their processing flow or physical characteristics. They involve representing the entire information content of the database in abstract terms relative to the way the data is physically stored (Batra et al. 1992). Therefore, developing CDM based on system requirements is a critical and demanding task in the overall database design.

Figure 2 represents the CDM of the ECDSWC data portal. It discusses entities, entity attributes, and relationships between entities. There are 16 entities generated and two types of cardinalities formed, which are used to represent the degree of relationships between entities, such as One (1) to Many (*) and Many (*) to Many (*). As result, the CDM show the data fragmentation strategies and their relationships in the database. The parent object **internal** entity is created to represent all the corporation data. From the **internal** entity, the **project_data** entity is formed that is the target data that can be managed by the data portal. This creates one-to-many relationships between the **internal** and **project_data** entities.

As the next step, the **project_data** entity is decomposed into **data_types_inv** and **doc_cat** entities and created one-to-many relationship on both sides. Both the **data_types_inv** and **doc_cat** entities are used to store a raw data's thematic area description and project report's category annotation respectively. The **grant_project** entity is formed by creating a bridge table between the **project_data** and **login11** entities, using many-to-many relationship. The **login11** entity is used to hold user information for the purpose of managing the data portal access

privileges and acquires user information from the corporation's LDAP server. Users are managed in groups; therefore, a **national_cor_center** entity is created to group user information under centers or sectors of the corporation. Sectors or centers are further grouped into different types of core processes. Hence, to represent the core process of the center or sector, the **eco_process** entity is created, which forms a one-to-many relationship with the **national_cor_center** entity.

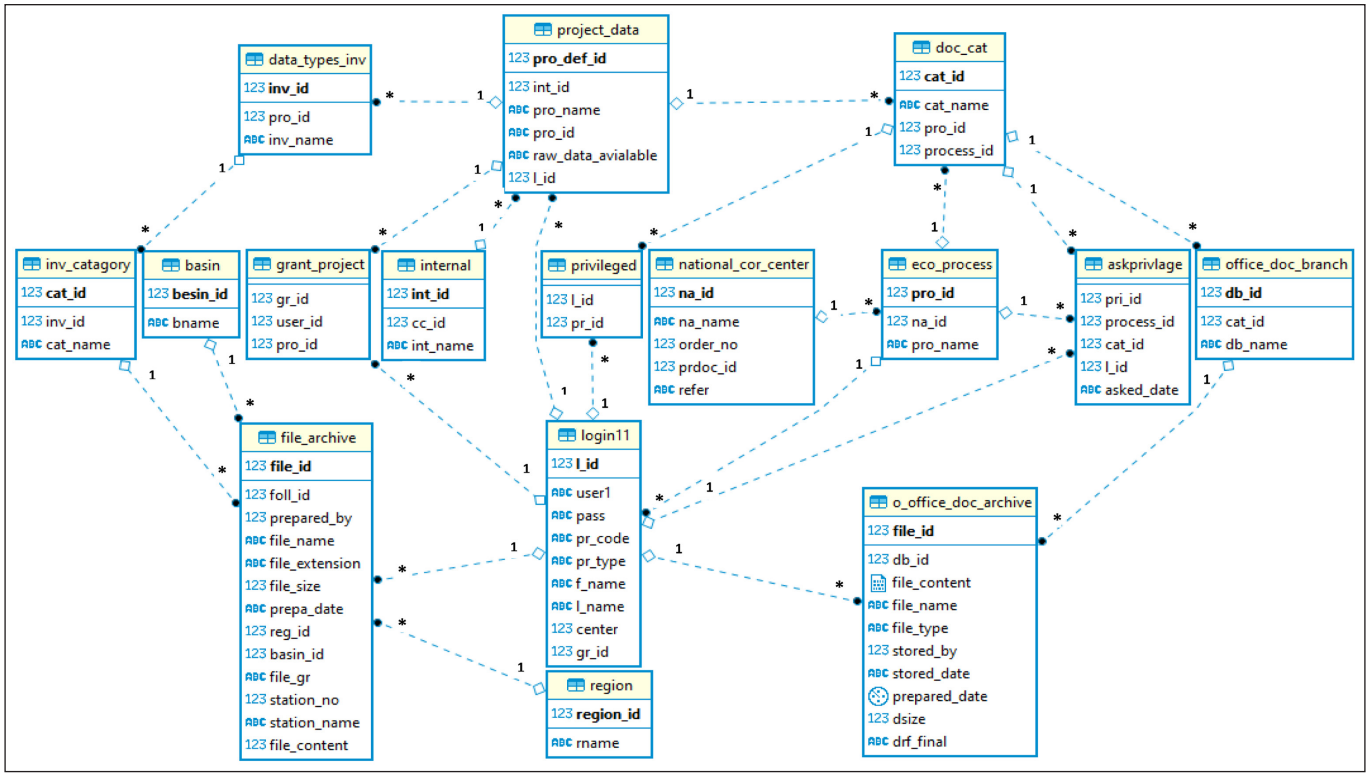


Figure 2 Conceptual data model.

The project report category, the **doc_cat** entity, is preserved by the core process of sectors or centers, which creates one-to-many relationships with **eco_process** entity. In addition, the **eco_process** and **login11** entities created one-to-many relationships that indicate the users of core process have information stored in the **login11** entity. Therefore, to access the project report category, the user must ask for privilege from the core process, which creates a bridge table called **ask_privilege** that forms many-to-many relationships between the **eco_process** and **doc_cat** entities, as well as an equivalent relationship between the **eco_process** and **login11** entities. Then the requested project report document is awarded to the user who made the request, which creates many-to-many relationships by forming a bridge table **privileged** in between the **doc_cat** and **login11** entities.

The thematic area of raw data, **data_type_inv** entity, is further decomposed into classification and creates an **inv_category** entity by using one-to-many relationship. The **inv_catagory** entity is used to store the classification metadata of the project's raw data. It creates a one-to-many relationship between raw data's binary file storage, which is **file_archive** entity. To store the raw data's binary file, the user needed to get the privilege, which creates a one-to-many relationship with the **login11** entity. Further, the **file_archive** entity creates a one-to-many relationship with the **region** and **basin** entities.

The other is the **office_doc_branch** entity, which stores sub-classified information about the project report's category metadata and creates one-to-many relationship with the **doc_cat** entity. Moreover, the **office_doc_branch** entity has one-to-many relationship with the **o_office_doc_archive** entity. The **o_office_doc_archive** entity is used to store project report documents. The user needs privileges to store the document; therefore, the **o_office_doc_archive** entity creates a one-to-many relationship with the **login11** entity.

2.4. PERFORMANCE TUNING

Previous studies have demonstrated the key considerations for database performance tuning, such as the database design principles, database query, I/O optimization, and parallel query

process. The goal of database performance tuning is to minimize the response time of queries (Kamatkar et al. 2018). This goal can be achieved by understanding the logical and physical structure of data, applications used on the system, and how the conflicts are managed on the database (Kamatkar et al. 2018).

2.4.1. Database design principle

Poor database design leads to inadequate database performance (Kamatkar et al. 2018) as it miscarries the database Atomicity, Consistency, Isolation, and Durability (ACID), which makes database lose durability, and makes it expensive when carrying out the operation. However, a good database design minimizes redundancy and anomalies, preserves known functional dependencies, and prevents spurious information from emerging (Albaraka et al. 2018). Normalization is database design method to organize data in relations or tables following specific rules to reduce data duplication and ensure referential integrity (Fong et al. 2021). A normalized database can also have effects on improving maintainability and scalability (Albaraka et al. 2018). Hence, it reduces costs associated with insertion, deletion, and changing anomalies. Even minimizing a small amount of anomalies can result in significant cost minimization (Sobol et al. 1996), which improves database performance.

According to (Albaraka et al. 2018), achieving Third Normal Form (3 NF) for all the database tables is advantageous over a non-normalized database structure. Therefore, every database should be normalized to at least 3 NF, meaning the primary keys are defined and columns are atomic; therefore, there are no repeating groups, no partial dependencies, and no transitive dependencies (Albaraka et al. 2018). Figure 3 shows the database design structure of the ECDSWC data portal. It provides mass storage to store data, such as spatial data and its attributes along with metadata and other documents.

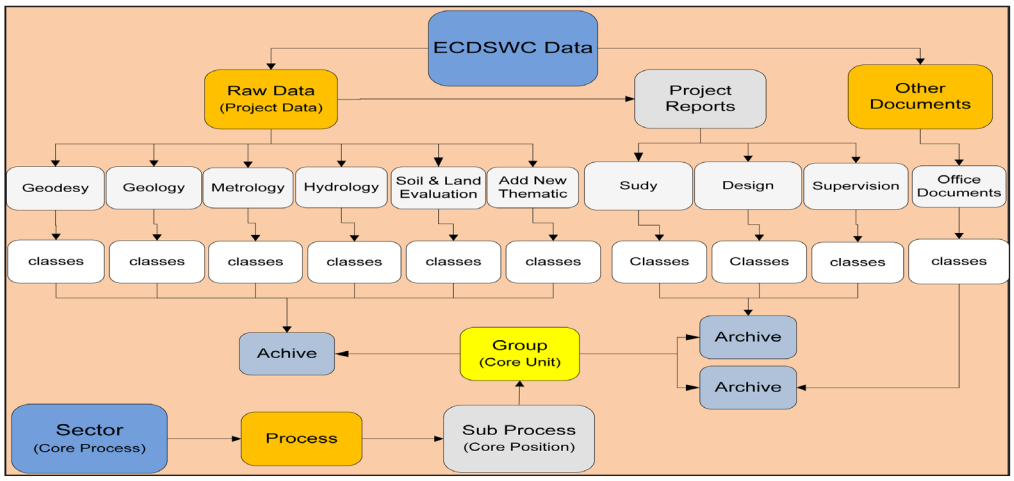


Figure 3 Data portal's database structure.

This novel design structure is achieved based on the principle of 3 NF. As stated in (Albaraka et al. 2018), the database design based on 3 NF meets all the requirements of 2 NF and removes transitive dependency from a unit database table, that is, all of the columns must be dependent on the primary key of the table, which improves database performance like flexibility, maintainability, and stability (Sobol et al. 1996). Apart from performance improvement, this database design ensures that transactions will not be affected by any other concurrent transaction; therefore, information saved in the database is immutable until another update or deletion transaction affects it. It is also critical to keep data transactions atomicity to ensure database system's reliability and if failure happens, it returns all the data to the state before the transaction was executed.

2.4.2. Database indexing

The maximum computation cost in the DBMS engine is fetching of information (Myalapalli et al. 2015). Indexing reduces query time in the database (Colley et al. 2017). It minimizes the number of disk accesses required when a query is processed. However, an improper or missing index is poor database index, and frequently updates database tables; therefore, it leads to

increase disk input/output (I/O) and memory wastage (Kamatkar et al. 2018). However, a proper index removes duplicates that affects system resources (Kamatkar et al. 2018). It is also recommended that indexing should be created on the accurate columns of the database tables (Sultana et al. 2017), like the primary key of the table.

PostgreSQL provides several indexing techniques, such as Balanced tree (B-tree), Hash, Generalized Index Search Tree (GiST), Specially Partitioned-GiST (SP-GiST), Generalized Inverted Index (GIN) and Block Range Index (BRIN) (Sultana et al. 2017). B-tree and hash are the two most used methods (Qu et al. 2019). The ECDSWC database is intended to use the B-tree index type as suggested in (Qu et al. 2019) for the range query process, that is, lower and upper limiting query when conducting data searches from database tables. As discussed in 2.3 and 2.4.1, the ECDSWC database structure is a complex database. Therefore, when creating indexes in the ECDSWC database structure, we selected three (3) database tables that were planned to contain millions of records. Further, the CONCURRENTLY keyword is used with the index query to avoid locking between multiple database operations. Moreover, creating an index yields extra structure in the database (Qu et al. 2019), holds additional space on the database, which creates tension on the overall system. To avoid this tension, the ECDSWC database used WHERE condition to disallow index creation when the item stored in the database is less than a certain number of records.

2.4.3. Parallel query processing

Parallel query processing enables system memories and processors (CPUs) to cooperatively work together to improve system performance in a data-intensive operation of the large database environment (Mancini et al. 2022). When the query optimizer decides to execute parallel query process in PostgreSQL, it creates a query planner. Then the query planner generates a parallel query plan. The process will request a number of background worker processes, which is equal to the number of workers chosen by the planner. The background workers that the planner uses are limited to at most `max_parallel_workers_per_gather`, that is, it depends on both `max_worker_processes` and `max_parallel_workers` (Schönig 2019). Therefore, it is possible for a parallel query to run with fewer workers than planned. The above commands such as `max_worker_process` display the number of available worker process during execution; `max_parallel_workers` display the number of workers available for parallel query and `max_parallel_workers_per_gather` identifies the total number of background worker when parallelism executes.

PostgreSQL executes several parallel query types including parallel sequential scan, parallel index scan, B-tree creation, aggregation and append (Schönig 2019). The data portal implements parallel B-tree index formation and parallel index scanning to reduce index creation and scanning time. As discussed in section 2.4.2, indexes speed up database searching processes. However, creating an index and scanning the index within a database that stores large datasets can still be computationally expensive.

2.4.4. JSON file transaction

XML and JSON are two data serialization formats used for data transmission between web applications (Afsari et al. 2017). XML produces child nodes and parent nodes while JSON uses lightweight data based on key-value pairs, which is effective to improve system performance (Yahui 2012). Therefore, it saves execution space and time, and reduces network latency. JSON file format has been widely used in web applications (Daquino et al. 2022; Ramachandran et al. 2018; Zioti et al. 2022; Wu et al. 2022). According to Ramachandran et al. (2018) it is best practice to use JSON file format for data sharing in repositories that have large-scale data. The ECDSWC data portal thus implements the **com.google.gson.Gson,** maven dependency, which encodes data in a JSON file format. On the implementation, the function `json.toJson<data_<param1,param2>>` calls the **data_<param1, param2 >** method to invoke a Data Access Object (DAO) that returns value into Map. Entry<key, value> pairs within JSON file format. The return value is then passed to the client side, which is the presentation layer.

2.4.5. Thread pool to handle user's query

The data portal is implemented thread pool with a maximum number of threads, which is available in the processor of the system hardware. In the implementation, the thread pool is

created on top of SQL batch processing, which handles bulk data and other user queries in the style of the multithreaded query handling structure. Instead of processing data sequentially in the loops by using single thread, allowing high number of cores for multithread is efficient (Schönherr et al. 2011). To do that the Java class object `<thread_pool>` is implemented.

Thread pool implementation in multi-tasking environment provides load balancing when the thread process shares mutual queue. To balance load and jobs in a queue, this study provides balanced file handling method by implementing java list object, `List<thread_pool>threads = Arrays.asList<param>,` which runs in the thread pool. Sometimes there will be a high thread demand that can kill active processes. Therefore, the thread balancing is used to optimize high thread demands on such occasion, which improves performance of the existing system.

The user queries and bulk data of different file format is then passed to a prepared statement, which executes a database query. Prepared statement creates a copy of the embedded SQL statement to iteratively execute Data Manipulation Language (DML) for the multiple sets of values. When the iteration ends, the `executeBatch()`method is called and executed to store data in the database.

3. RESULTS AND DISCUSSIONS

3.1. DATA PORTAL MODULES

The data users that use large and broadly classified data can take the advantages of data portal features to store, manage, and share large-sized binary data of different file formats. In addition, the data portal provides UIAS features that consolidate security services and manage data users. UIAS enables users to log into the data portal using active directory domain services accounts of ECDSWC. When users log into the system, a secured connection is established between client and server, and provides Read-Only and Read-Write privilege to the users.

3.1.1. File upload module

Figure 4 shows user interface (GUI), which is granted for an 'Elite User' who has Read-Write (Wr') privilege. The user is responsible for storing and managing files in the repository. When the User clicks on the 'Add Project Data' node in the GUI (Figure 4), the 'Add Project's Raw Data' sub-module is displayed on the main window. In this sub-module, the user can add single or multiple files, which are then segmented and annotated with metadata. The contents of the interface shown in Figure 4 are a bulky and broadly classified dataset as well as metadata, ready to upload when the user presses the upload button. A check box and a drop-down list are used to segment and annotate files with their metadata descriptions.

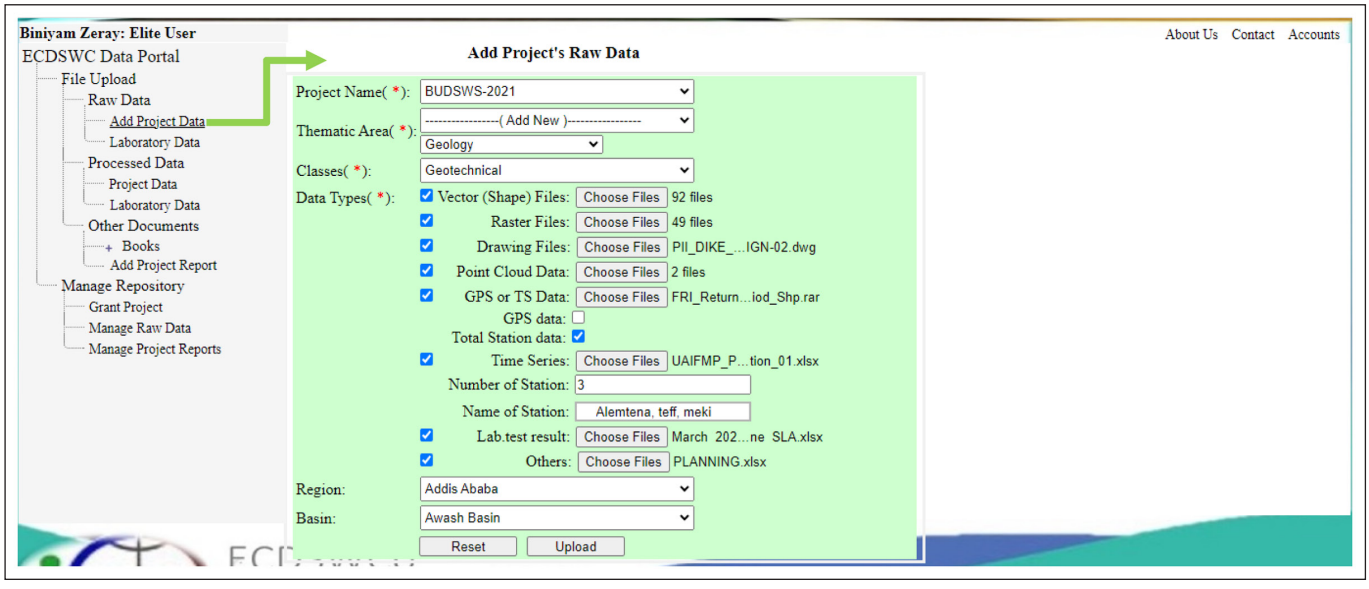


Figure 4 Raw data uploading module.

Based on the descriptions of metadata, N numbers of files with multiple categories can be added to the interface. At the end, datasets and their metadata are simultaneously

sent to the service layer. The service layer handles the large-sized bulk datasets and their metadata by using thread pools. Thereafter, the thread pool shares task loads to minimize computational complexity. Many data management applications create new threads redundantly to handle data and user queries, which creates performance overhead. To overcome the issue, the modular data portal implements thread pooling to share the single thread operation's tension in between threads. In addition, the data portal avoids process deletion between threads by balancing the load and queueing jobs. Therefore, thread pooling significantly enhances computational overhead.

When discussing this use cases: **Add Projects Raw Data Sub-Module,** the goal of the user is to upload raw data. The raw data is collected from different projects launched by the ECDSWC and uploaded into the repository. To upload the dataset; first, the dataset or binary file shall be buffered into the <Add Project's Raw Data> module. Then the user has to embed annotation or metadata such as project name and thematic areas from the drop-down list, and then datasets are segmented under the data type, which is controlled by a checkbox. The dataset's thematic area and possible classes are as follows. For all categories, an 'Add New' option is available.

- Hydrology: cascaded into classes of stream flow and water level.
- Geology: cascaded into classes of Geotechnical, Geophysics, and Geochemistry.
- Geodetic: cascaded into classes of LULC, Land Surveying, and Arial Surveying.
- Meteorology: cascaded into classes of Rainfall, Solar Radiation, Temperature, Humidity and Wind.
- Soil and Land Evaluation: cascaded into classes of Auger, Profile, Insitu Measurement, Soil Lab Data, and Land Suitability Evaluation.

Further, checkboxes are used to define data types, which are under the classification of the dataset. Therefore, the user checks the checkbox to annotate data types of the binary data before storing data in the repository.

The other use case we discuss here: **Add Project Report Sub-Module**. When the user clicks on this node from the parent tree of GUI (Figure 4), the sub-module is displayed on the main window as shown in the Figure 5. The user adds file or binary data to this sub-module and then metadata is annotated from a drop-down list, which is a technical definition of the project report and stored in the repository. The project report is the working document or results derived from project's raw data. Finally, the project report and its metadata are uploaded into the file repository when the user hits an 'Upload' button. Note, if the project reports do not have the related raw data in the repository, the user needs to provide technical definitions and then annotate new metadata such as project name and ID. This project report is considered confidential data and managed under the centers/sectors of ECDSWC. The project reports are grouped into the following categories. Again, an 'Add New' option is provided for all categories.

- Design: cascaded into Design work, initiation document, and inception document.
- Supervision: cascaded into initiation document, inspection document, compilation document.
- Study: cascaded into Feasibility Study.

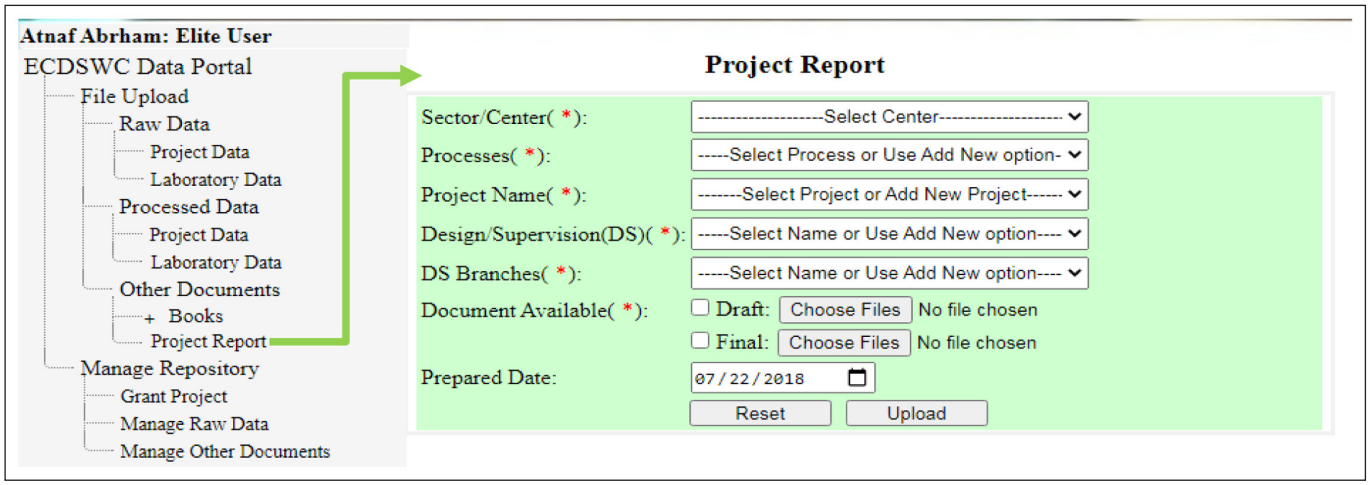


Figure 5 Row data's report upload module.

Within the **Manage Repository Module**, files are managed in repository, with respect to the integrity policy. The privileged 'Elite User' can update annotations and other metadata for a binary file. In this context, the privileges granted to the 'Elite User' include the following:

1. Grants project access: Users who created the project can share his/her projects to others.
2. Manages dataset in the repository (raw data or project report): Users are responsible and accountable for the data they stored and if necessary, they can remove the files or binary data from the repository.

3.1.2. Search engine

A search engine provides search execution that allows sophisticated filtering and discoveries. It allows users to start searches using metadata or annotation of files or binary data descriptions. When the user sends a search parameter or keywords of metadata from the GUI, the data access object is invoked. Based on the search key, the annotations of binary data are processed in the file repository and then a result is sent to the presentation layer via the controller model. The raw data is retrieved with the perspective of thematic area of the study such as geodesy, meteorology, hydrology, geology, and soil and land evaluation. We have two options to access datasets such as Global Search and Table-Based Search. The global search engine uses keywords from the project names (Figure 6). Based on the keyword given, all annotated names for the project data are displayed and then the user chooses the relevant project name of the project to display document or binary data under its thematic area.

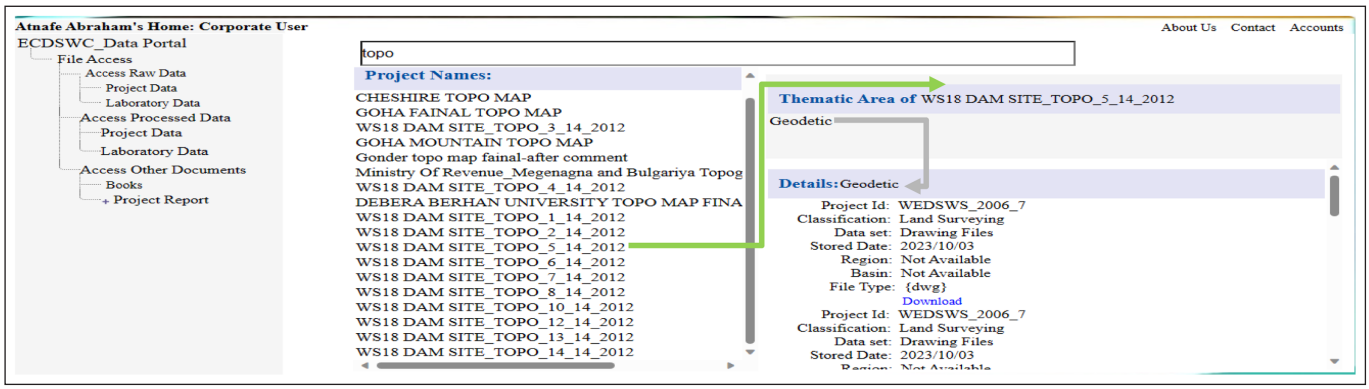
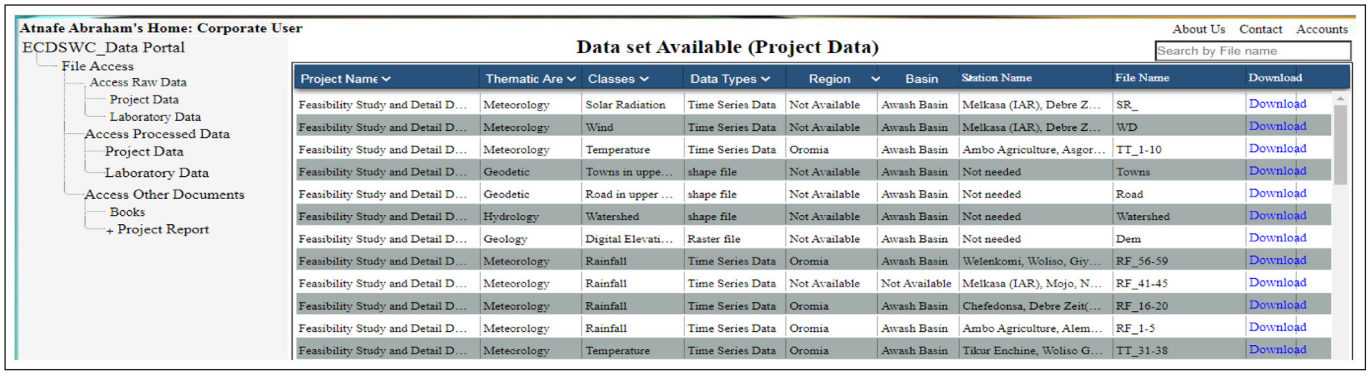


Figure 6 Global search Interface.

The table-based search lists all the annotations of the binary data from the repository (Figure 7). The user can then filter to find the target binary data. Metadata, which is populated in the table header, is used to filter binary data on the table. Therefore, the user filters the table by using the project name from the metadata, then the thematic area of datasets, followed by datasets classification, as seen from Figure 7. For instance, the user selects project name by clicking on the <Project Name> drop down list, then thematic area drop-down list to select the thematic area of the project (Hydrology, Geology, Geodetic, Meteorology, and Soil and Land Evaluation). Finally, the user hits on the classes of thematic area, which display the targeted binary data. At the end, the user downloads the directed file.

Figure 7 Table-based dataset access.



3.2. TESTING AND EVALUATION

3.2.1. Experiments and test cases

The modality, functionality, and structure of the data portal, as well as cardinality, confidentiality, and integrity of the data, were tested based on the experiments done. Tests were carried out from both users' and experts' perspectives. From a user perspective, the data portal was tested to examine the ease and multi-task supportiveness of the GUI when accessing, storing, and sharing datasets. The test revealed that the GUI can handle concurrent and simultaneous operations, is flexible and interactive, and can handle multiple types of requests and responses via multitasking. Further, it is structurally dynamic because of a complete implementation of hypermedia; as a result, the system loads data or mapping responses on the current page rather than redirecting to the new page. The tests also confirmed that the data portal is reliable and efficient, as measured by the number of failing cases from the total number of test cases performed when storing, managing, and sharing data simultaneously between users. On the failure measurement test, the system was tested via five users with different types of concurrent and simultaneous operations. The first user performed metadata annotation, uploaded raw data, project reports, and managed data and its metadata. The second user completed metadata annotation and uploaded a project's raw data. The third user worked on searching and viewing of metadata about datasets and then downloaded datasets. The fourth and fifth users did requests for the project report and granting operation, respectively. All the operations were performed in a concurrent manner and simultaneously, which is a total of ten operations. From these operations, zero failures and ten successful operations are recorded. Therefore, it is evident that the data portal is 100 percent efficient and reliable under these test conditions.

From the expert perspective, a test was done in order to assure functionality, availability, and accuracy of the system. A test was also done to check the integrity and privacy of the data in the repository, as well as to teste a transaction commit and rollback. These test cases used single and bulk modes of Create, Read, Update, and Delete (CRUD) concurrent operations. Then the ACID, data transaction mechanism was measured to evaluate the degree of changes that occurred to the existing transaction when other concurrent transactions took place. The test was performed by User 1 as well as User 2. The test revealed that transactions were never affected by any other concurrent transaction; therefore, data and its metadata in database are immutable until another update or deletion operation affects it.

The other test case used in the experiment was to verify the performance of the data portal. Experiments were done with a set of input variables within group of five different bulky datasets, such as Meteorology, Geology, Hydrology, Geodetic, and Soil and Land evaluation. In each group, more than five users concurrently perform their operations using CRUD operations. In this test case, file downloading speed of 12 MB/s was recorded when User 3 simultaneously downloaded two different bulky datasets. This shows that the data portal has high performance capacity, and the privacy of data is preserved. Toward this end, this modular data portal adopted the: – AJAX mechanism, REST API principle, database optimization, and thread pooling to its implementation. HTTP requests and responses are exchanged between client and server in the process of resource mapping, which means the data loading and replying to the request are done through multiple endpoints in an asynchronous manner (Lawi et al. 2021). In the REST API, there is no code compilation or page redirection computation cost; therefore, the hyperlink decoration mechanism is independent from the resource identification schema (Li et al. 2016) and defined independently from the schema. Conversely, when the system uses URL redirection, it creates new web pages to handle HTTP responses. As a result, it requires more computational time for code compilation and page creation when compared with loading or mapping of data on the current page, due to DNS lookup time, TCP connection time, SSL handshake time, HTTP header elements, page download time, number of DNS iterations, and number of HTTP redirects (Asrese et al. 2016).

These experiments and test cases were based on 352 projects data stored in the repository, which contained 3,318 binary files of different file formats (see Figure 8 (a) and (b)). Moreover, the binary files in the database add up to around 69 GB in size, as shown in Figure 9.

Generally, these experiments revealed that the data portal can support concurrent operations with acceptable performance, delay tolerance, and keeps privacy and integrity of the data. Therefore, the experiments and test cases showed that this data portal is secured and modular,

and capable and robust to store, manage, and share large-sized datasets. It is also capable of controlling concurrent operations and able to manage all the users and workgroups that can access the system.

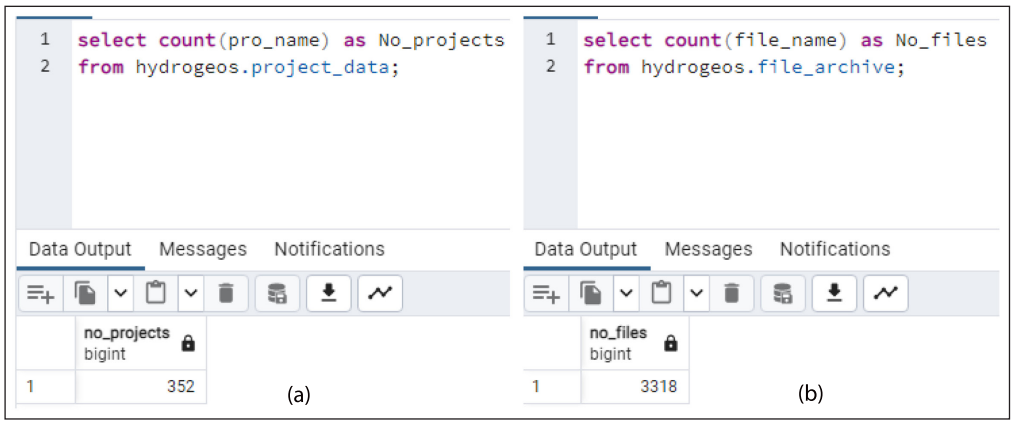


Figure 8 (a) Query that counts projects data stored in database. **(b)** Query that counts binary files stored in database.

Figure 9 Query that sums file size of raw data in gigabytes.

3.2.2. System evaluation

Normalization and de-normalization of databases are the two inverse activities. Some prior research argues that normalization, such as 3 NF, augments databases by adding additional tables and requiring joins in queries, making it more expensive than a non-normalized database. Even though there will be more tables after normalization, joining tables is faster and more efficient because it sets a small number of parameters; therefore, the queries will be less complicated compared to the non-normalized design (Albaraka et al. 2018). Moreover, weakly normalized or non-normalized tables can store a larger number of files compared to normalized designs due to potential for data redundancy. As a result, it may increase record size and overhead I\O computation (Sobol et al. 1996). Therefore, implanting 3 NF in this study kept all the columns atomic, and reduced repetition, partial dependencies, and transitive dependencies. As a result, the 3 NF improves the database query performance when comparing with a non-normalized database.

Regarding database indexing, B-tree minimizes computation during the disk's I/O operation. As the analysis presented in Figure 10 and Table 1 show, the computational time of B-tree is O($logn$) while sequential scan takes O(n). The analysis showed that the running time of sequential scan grows faster than B-tree index. We carried out experiments on the data portal using the aforementioned 352 ECDSWC projects, which contains 3,318 different types of binary data (Figure 8 (a) and (b)), and measure about 69 GB in size (Figure 9).

Table 1 summarizes the query execution time of both sequential and B-tree indexes. We observed that database operation with B-tree is computationally much better than sequential operation. Other measurements adopted on the data portal were parallel query processing, accessing JSON file formats, and thread pooling. Parallel query processing reduced the index

creation and table scanning time. Both thread pooling and accessing JSON file formats added value to reduce computational time and network latency.

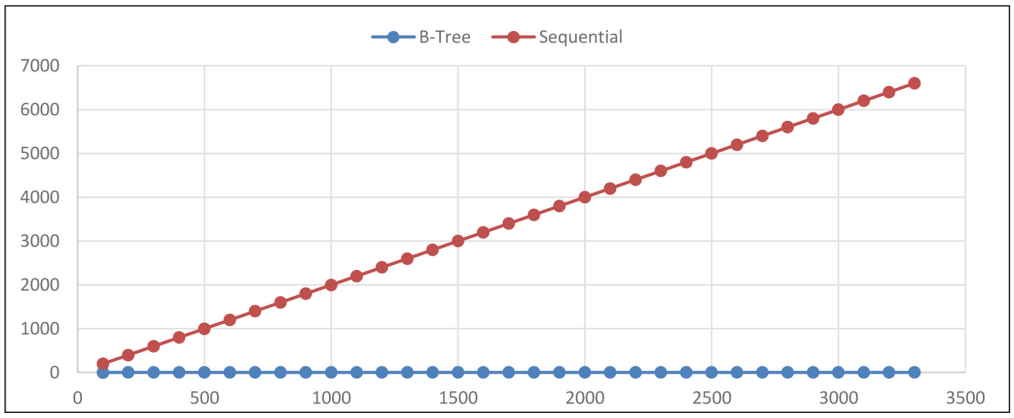


Figure 10 B-Tree and full table scan performances analysis.

OPERATIONS/QUERIES	B-TREE			FULL TABLE SCAN		
	BEST CASE	AVERAGE	WORST CASE	BEST CASE	AVERAGE	WORST CASE
Space	O(1)	O(*n*)	O(*n*)	O(1)	O(1)	O(1)
Search	O(1)	O(log *n*)	O(log *n*)	O(1)	O(n/2)	O(n)
Insert	O(log *n*)	O(log *n*)	O(log *n*)	O(n)	O(n)	O(n)
Delete	O(log *n*)	O(log *n*)	O(log *n*)	O(n)	O(n)	O(n)

Table 1 B-tree and full table scan comparision.

4. CONCLUSIONS

A modular and secured data portal was developed based on demand for management of knowledge and data in ECDSWC. The demand assessment was done using questionnaires to representative professionals of ECDSWC. The source of primary data for this data portal are experts, work groups, and project teams of multiple sectors. The assessment given us insights to design and implement data portal, and helped us to understand data storing, managing, and sharing uncertainties faced in ECDSWC. It also helped us to identify the thematic areas of dataset, types of documents, and binary data formats used in ECDSWC. Further, the preliminary assessment gave us a hint on the weaknesses pertaining to data integrity, privacy, and data-storage management in the ECDSWC.

Based on the assessments done, a REST API principle-based data portal was implemented to manage large-sized and heterogeneous data of different file formats and corresponding metadata. On top of data management, the system ensures integrity, privacy, and quality of data, and improves performance. The modality, cardinality, functionality, and structure of the data portal were tested and showed that the portal's GUI is simple to use and supports multi-tasking. The repository of the data portal is also robust enough to store, manage, access, and share a variety of datasets. The design principle, that is, abstraction, modularity, extensiveness, and maintainability of the data portal, were also tested based on community demonstrated approaches. The tests demonstrated the system's modularity, reliability, and efficiency. In handling concurrent and simultaneous operation, the tests measured a 0 percent failure and 100 percent successful operation rate. The experiments also revealed that data portal performs at a high speed of about 12 MB/s uploading and downloading time.

Even though the data portal is reliable, efficient, robust, multi-tasking, and high-performing, the following research gaps will be addressed in the future implementation:

- Investigation of positive and negative impacts of database normalization, specifically debates about the design principle, which includes normalized or non-normalized database. The current implementation of the data portal adopted the 3 NF of database

design; however, in future implementation, this will be analyzed and proven by developing an algorithm.

- The future data portal will implement and integrate a real time data collection module. This module will have a field data acquiring functionality, that is, data sheets and real-time data quality checking functions.
- The future data portal will also enfold data integration and intelligent data-processing functions to achieve the optimum exploitation and visualization of knowledge and insights from heterogeneous datasets. In this context, the operations of intelligent data processing will be used for identification and prediction of data structure and anatomy. Then, the classified and transformed digital map will be used to demonstrate the dynamics of diversified data of points or grids.

FUNDING INFORMATION

This work is supported by Ethiopian Construction Design and Supervision Works Corporation; Surveying, Geospatial and Civil Informatics Center; Addis Ababa Ethiopia.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Atnafu Abrham Lencha orcid.org/0000-0002-9491-3059
Surveying, Geospatial and Civil Informatics Center, Ethiopian Construction Design and Supervision Works Corporation, Addis Ababa, Ethiopia

Addisalem Bitew Mitiku orcid.org/0000-0002-5654-1717
Surveying, Geospatial and Civil Informatics Center, Ethiopian Construction Design and Supervision Works Corporation, Addis Ababa, Ethiopia

Abel Tadesse Woldemichael orcid.org/0000-0003-2530-891X
Surveying, Geospatial and Civil Informatics Center, Ethiopian Construction Design and Supervision Works Corporation, Addis Ababa, Ethiopia

REFERENCES

Afsari, K, Eastman, CM, and **Lacouture, CD** 2017. JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. *Automation in Construction International Journal,* 77: 24–51. DOI: https://doi.org/10.1016/j.autcon.2017.01.011

Albaraka, M and **Bahasoon, R** 2018. Prioritizing technical debt in database normalization using portfolio theory and data quality metrics. In: *Int. Conference on Technical Debt (TechDebt '18), Association for Computing Machinery (ACM)*. pp. 31–40. DOI: https://doi.org/10.1145/3194164.3194170

Amugongo, L, et al. 2016. Open data portal – A technical enabler to drive innovation in Namibia. In: *2nd Int. Conf. on Open and Big Data (OBD),* Vienna, Austria. pp. 80–86. DOI: https://doi.org/10.1109/OBD.2016.19

Asrese, AS, et al. 2016. *A tool for automated web performance measurement.* IEEE Globecom Workshops (GC Wkshps), Washington, USA. pp. 1–6. DOI: https://doi.org/10.1109/GLOCOMW.2016.7849082

Batra, D and **Davis, JG** 1992. Conceptual data modelling in database design: Similarities and differences between expert and novice designers. *Int. Journal of Man-Machine Studies,* 37(1): 83–101. DOI: https://doi.org/10.1016/0020-7373(92)90092-Y

Bauermeister, S, et al. 2020. The Dementias Platform UK (DPUK) Data Portal. *European Journal of Epidemiology,* 35: 601–611.

Breunig, M, et al. 2020. Geospatial data management research: Progress and future directions. *ISPRS International Journal of Geo-Information,* 9(2): 95. DOI: https://doi.org/10.3390/ijgi9020095

Colley, D and **Stanier, C** 2017. Identifying new directions in database performance tuning. *Procedia Computer Science,* 121: 260–265. DOI: https://doi.org/10.1016/j.procs.2017.11.036

Daquino, M, et al. 2022. Creating RESTful APIs over SPARQL endpoints using RAMOSE. *Semantic Web,* 13(2): 195–213. DOI: https://doi.org/10.3233/SW-210439

Divac, D, et al. 2009. Hydro-information systems and management of hydropower resources in Serbia. *Journal of the Serbian Society for Computational Mechanics,* 3(1): 1–37.

Fong, JS and **Yan, KWT** 2021. *Data normalization. Information systems reengineering, integration and normalization.* Springer. pp. 343–376. DOI: https://doi.org/10.1007/978-3-030-79584-9_8

Halili, F and **Ramadani, E** 2018. Web services: A comparison of soap and rest services. *Modern Applied Science,* 12(3): 175. DOI: https://doi.org/10.5539/mas.v12n3p175

Islam, M, et al. 2020. Coding practices and recommendations of spring security for enterprise applications. In: *2020 Conference of IEEE Secure Development*. pp. 49–57. DOI: https://doi.org/10.1109/SecDev45635.2020.00024

Ji, Y, et al. 2019. Multi-thread concurrent compression algorithm for genomic big data. In: *Int. Conf. on Parallel & Distributed Compute*. pp. 475–478. DOI: https://doi.org/10.1109/PDCAT46702.2019.00093

Kamatkar, SJ, Kamble, A, et al. 2018. Database performance Tuning and query optimization. *Data Mining and Big Data, Lecture Notes in Computer Science,* 10943: 3–11. DOI: https://doi.org/10.1007/978-3-319-93803-5_1

Kumari, S and **Rath, SK** 2015. Performance comparison of SOAP and REST based web services for enterprise application integration. In: *Int. Conference on Advances in Computing, Communications and Informatics (ICACCI)*, India. pp. 1656–1660. DOI: https://doi.org/10.1109/ICACCI.2015.7275851

Lawi, A, et al. 2021. Evaluating GraphQL and REST API services performance in a massive and intensive accessible information system. *Computers,* 10(11): 138. DOI: https://doi.org/10.3390/computers10110138

Li, L, Chou, W, Zhou, W, and **Luo, M** 2016. Design patterns and extensibility of REST API for networking applications. *IEEE Transactions on Network and Service Management,* 13(1): 154–167. DOI: https://doi.org/10.1109/TNSM.2016.2516946

Li, M and **Luo, N** 2009. Data sharing between web applications based on the request of user. In: *2009 ISECS Int. Colloquium on Computing, Communication, Control, and Management*, Sanya, China. pp. 280–282, DOI: https://doi.org/10.1109/CCCM.2009.5268058

Lnenicka, M and **Nikiforova, A** 2021. Transparency-by-design: What is the role of open data portals. *Telematics and Informatics,* 61(101605): 0736–5853. DOI: https://doi.org/10.1016/j.tele.2021.101605

Mancini, R, et al. 2022. Efficient massively parallel join optimization for large queries. In: *SIGMOD 22 Proceedings of the 2022 Int. Conference on Management of Data, ACM*. pp. 122–135. DOI: https://doi.org/10.1145/3514221.3517871

McGovern, J, Tyagi, S, Stevens, M, and **Matthew, S** 2003. *Java web services architecture*. 1st Edition. Morgan Kaufmann. DOI: https://doi.org/10.1016/B978-155860900-6/50003-8

Mitiku AB, et al. 2020. The need for integrated flood management approach from social, economic and environmental perspectives: The case of Upper Awash River Basin. *International Journal of Water Resources and Environmental Engineering, Academic Journals,* 12(3): 57–70.

Myalapalli, VK, et al. 2015. Augmenting database performance via SQL tuning. In: *2015 Int. Conference on Energy Systems and Applications*, Pune, India. pp. 13–18. DOI: https://doi.org/10.1109/ICESA.2015.7503305

Nguyen, Q and **Baker, O** 2019. Applying spring security framework and OAuth2 to protect micro service architecture API. *Journal of Software,* 14(6): 257–264. DOI: https://doi.org/10.17706/jsw.14.6.257-264

Pandey, AK, et al. 2020. Key issues in healthcare data integrity: Analysis and recommendations. *IEEE Access,* 8: 40612–40628. DOI: https://doi.org/10.1109/ACCESS.2020.2976687

Pereira, J, et al. 2022. A platform for integrating heterogeneous data and developing smart city applications. *Future Generation Computer Systems,* 128: 552–566. DOI: https://doi.org/10.1016/j.future.2021.10.030

Qu, W, et al. 2019. Hybrid indexes by exploring traditional B-tree and linear regression. In: *Int. Conference on Web Information Systems and Applications*. pp. 601–613. DOI: https://doi.org/10.1007/978-3-030-30952-7_61

Ramachandran, GS, et al. 2018. Towards a decentralized data marketplace for smart cities. In: *IEEE Int. Smart Cities Conference (ISC2), Kansas City*, MO, USA. pp. 1–8. DOI: https://doi.org/10.1109/ISC2.2018.8656952

Schönherr, M, et al. 2011. Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs. *Computers and Mathematics with Applications,* 61: 3730–3743. DOI: https://doi.org/10.1016/j.camwa.2011.04.012

Schönig, H-J 2019. *Mastering PostgreSQL 12*. 3rd Edition. Packt.

Sherman, R 2015. *Foundational data modelling. Business intelligence guidebook*. Morgan Kaufmann. pp. 173–195. DOI: https://doi.org/10.1016/B978-0-12-411461-6.00008-3

Sobol, G, Kagan, A, and **Shimura, H** 1996. Performance criteria for relational databases in different normal forms. *Journal of Systems and Software,* 34: 31–42. DOI: https://doi.org/10.1016/0164-1212(95)00062-3

Sultana, S and **Dixit, S** 2017. Indexes in PostgreSQL. In: *Int. Conf. on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bengaluru, India. pp. 512–515. DOI: https://doi.org/10.1109/ICIMIA.2017.7975511

Wang, H and **Gong, C** 2016. Design and implementation of unified identity authentication service based on AD. In: *2016 8th Int. Conference on Computational Intelligence and Communication Networks (CICN)*, Tehri, India. pp. 394–398. DOI: https://doi.org/10.1109/CICN.2016.84

West, M 2011. *Some types and uses of data models. Developing high quality data models*, R. Adams and D. Bevans, (eds.), USA: Morgan Kaufmann. Available: ScienceDirect. DOI: https://doi.org/10.1016/C2009-0-30508-5

Wu, J, Orlandi, F, et al. 2022. Link climate: An interoperable knowledge graph platform for climate data. *Computers & Geosciences*, 169(105215). DOI: https://doi.org/10.1016/j.cageo.2022.105215

Wu, Z, Huang, W and **Yu, L** 2014. Design and implementation of unified identity authentication system Based on LDAP in Digital Campus. *Advanced Materials Research*. pp. 1213–1217. DOI: https://doi.org/10.4028/www.scientific.net/AMR.912-914.1213

Wu, Z, et al. 2020. An ontology-based framework for heterogeneous data management and its application for urban flood disasters. *Earth Sci Informatics*, 13: 377–390. DOI: https://doi.org/10.1007/s12145-019-00439-3

Yahui, Y 2012. Impact data-exchange based on XML. In: *7th Int. Conference on Computer Science & Education (ICCSE)*, Melbourne, Australia. pp. 1147–1149, DOI: https://doi.org/10.1109/ICCSE.2012.6295268

Zioti, F, et al. 2022. A platform for land use and land cover data integration and trajectory analysis. *International Journal of Applied Earth Observations and Geo Information*, 106(102655): 1569–8432. DOI: https://doi.org/10.1016/j.jag.2021.102655

Züfle, A, et al. 2020. Managing uncertainty in evolving geo-spatial data. In: *21st IEEE Int. Conference on Mobile Data Management (MDM)*. pp. 5–8. DOI: https://doi.org/10.1109/MDM48529.2020.00021

TO CITE THIS ARTICLE:
Lencha, AA, Mitiku, AB and Woldemichael, AT. 2024. Secured and Modular Data Portal: Database System to Manage Broadly Classified and Large-Scale Data. *Data Science Journal*, 23: 20, pp. 1–17. DOI: https://doi.org/10.5334/dsj-2024-020

Submitted: 16 January 2023
Accepted: 18 March 2024
Published: 15 April 2024

Data Science Journal is a peer-reviewed open access journal published by Ubiquity Press.