sustainability

MDPI

Article

Design and Implementation of a Scalable Data Warehouse for Agricultural Big Data

Asterios Theofilou [1,*], Stefanos A. Nastis [1], Michail Tsagris [2], Santiago Rodriguez-Perez [3] and Konstadinos Mattas [1]

[1] Department of Agricultural Economics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece; snastis@auth.gr (S.A.N.); mattas@auth.gr (K.M.)
[2] Department of Economics, University of Crete, 74100 Rethymno, Greece; mtsagris@uoc.gr
[3] Biotechnology Applications, IDENER, Early Ovington 24 Nave 8-9, 41300 Seville, Spain; santiago.rodriguez@idener.ai
* Correspondence: tasterios@agro.auth.gr

Abstract: The rapid growth of agricultural data necessitates the development of storage systems that are scalable and efficient in storing, retrieving and analyzing very large datasets. The traditional relational database management systems (RDBMSs) struggle to keep up with large-scale analytical queries due to the volume and complexity inherent in those data. This study presents the design and implementation of a scalable data warehouse (DWH) system for agricultural big data. The proposed solution efficiently integrates data and optimizes data ingestion, transformation, and query performance, leveraging a distributed architecture based on HDFS, Apache Hive, and Apache Spark, deployed on dockerized Ubuntu Linux environments. This paper highlights the reasons why a DWH is irreplaceable for big data processing, without disputing the strengths of traditional databases in transactional use cases. By detailing the architectural choices and implementation strategy, this study provides a practical framework for deploying robust DWH solutions that are useful in supporting agricultural research, market predictions and policy decision-making.

Keywords: data warehouse; big data; agricultural data

Academic Editor: Piotr Prus

Received: 7 March 2025
Revised: 14 April 2025
Accepted: 18 April 2025
Published: 20 April 2025

Citation: Theofilou, A.; Nastis, S.A.; Tsagris, M.; Rodriguez-Perez, S.; Mattas, K. Design and Implementation of a Scalable Data Warehouse for Agricultural Big Data. *Sustainability* **2025**, *17*, 3727. https://doi.org/10.3390/su17083727

1. Introduction

A diverse range of sources are constantly generating agricultural data. Climate models, market transactions, IoT sensors, satellite imagery, and supply chain records are only some of them [1]. The effective management and utilization of these data are crucial when attempting to make yield predictions, make price predictions, assess the climate risks, make policy decisions, or apply precision agriculture techniques [2]. For instance, in their study, Rana et al. [3] demonstrated that utilizing a big data framework significantly improved the prediction accuracy of agricultural commodity price forecasting. Traditional database management systems (DBMSs) cannot handle these massive and varied datasets. When data are in the range of petabytes, exabytes, or more, or in heterogeneous formats, DBMSs lack the distributed processing needed to cope. As data volumes grow, so does the need for a scalable framework, optimized for analytical queries.

A data warehouse (DWH) is designed to address these challenges. It provides structured and scalable high-performance mechanisms for the storage and retrieval of data, which are optimized for big data analytics [4]. Data can be processed in two distinct ways, with the first being the online transactional processing (OLTP) of data, for which relational

databases are optimized and best suited, and the other being online analytical processing (OLAP), which a DWH is built to support. While relational databases are irreplaceable for structured data and transactional processing, a DWH is the only practical solution for big data analytics. Since relational databases simply fail to handle datasets characterized as big data, our study highlights architectural choices rather than direct performance comparisons.

This paper presents the design and implementation of a scalable DWH for agricultural big data, emphasizing the distributed storage and processing, the performance optimization and the flexible deployment and scalability of the system, and finally, it addresses the data security and accessibility. This is achieved firstly by utilizing HDFS for scalable storage and Apache Hive for querying large datasets; secondly by implementing Apache Spark for in-memory processing that reduces the query latency compared to traditional MapReduce methods; thirdly by leveraging Dockerized containers on Ubuntu Linux, ensuring modularity and ease of deployment across cloud and on-premise environments; and fourthly by implementing role-based access control (RBAC) authentication and data governance policies to regulate user access.

Agricultural data are spatiotemporal, seasonal, and generated from heterogeneous sources. The proposed architecture was developed with these characteristics in mind. More specifically, HDFS enables the scalable storage of long-term time-series data, Hive can support structured analytics being performed in irregularly sampled records, while Kafka and Spark make possible the real-time processing of sensor streams. The proposed setup ensures that the DWH can handle the unique data-processing requirements of agricultural data.

The objective of this study is to present the design and implementation of a DWH solution that is tailored to agricultural data. Specifically, this paper aims to address the limitations that traditional relational database management systems (RDBMSs) face when handling diverse and large-scale agricultural data. To achieve this aim, this work will present the state-of-the-art software tools and their architectural orchestration, which ensure data security, enhanced query performance on diverse and massive data source inputs, extensibility and fault tolerance.

The rest of this paper is arranged as follows. Section 2 presents the findings of the literature review that guided the design decisions for the DWH implementation. In Section 3, the details of the proposed system are presented, including the data flows and management, architecture design, and security. Section 4 presents the results of our work, and Section 5 concludes with a discussion on the recommendations for agricultural big data management and considerations for future improvements.

2. Literature Review

In numerous studies, the development of data warehouse (DWH) architectures and extract transform load (ETL) processes has been extensively researched, and many aspects of the design, optimization, scalability, real-time processing, agricultural applications, and implementation have been investigated. This section presents some key research contributions that have guided the development of our scalable agricultural DWH system.

2.1. Data Warehouse Architectures and Optimization

A DWH needs to be set up in such a way that the software tools seamlessly integrate with each other. This arrangement of tools and software is the architectural design of the system. The following three studies guided our architectural choices and supported the integration of some of the Apache big data framework tools.

Regarding the choices that need to be made with respect to the architectural settings and data warehouse optimization, Yang, Ge and Helfert [5], after analyzing 116 publications and modeling 73 data warehouse architectures, identify 9 representative architectures that

they summarize into a "big picture", providing a comprehensive overview of the DWH architectures. In a same manner, Chaudhuri and Dayal [6] focus on the importance of online analytical processing (OLAP) systems in data warehouses, and they also discuss the relational OLAP (ROLAP), multidimensional OLAP (MOLAP), and hybrid OLAP (HOLAP) architectures. Their findings reinforce the need for distributed query processing abilities in a DWH, which we achieve through Apache Hive on Hadoop.

Jameel et al. [7], in their work, compare centralized vs. distributed DWH architectures, and they conclude that distributed architectures offer better fault tolerance and scalability. They also evaluate the ETL optimization techniques and find that distributed ETL pipelines (e.g., Apache Spark, Apache Flink, Apache Kafka) outperform traditional monolithic ETL processes. This further establishes our system's architecture, which uses Spark-based ETL for agricultural big data ingestion and transformation.

Thesma et al. [8] also support the use of distributed architectures in agriculture, proposing a Hadoop cluster for high-throughput cotton phenotyping. Their work highlighted the superior performance of the multi-node setup over the single-node systems and the value of scalable distributed systems for agricultural monitoring.

In their review, Cravero et al. [9] confirm that most agricultural data storage systems rely on Hadoop, Hive and Spark for distributed storage and large agricultural datasets, strengthening our argument for using the Hadoop ecosystem tools.

2.2. ETL Processes and Real-Time Data Warehousing

To manage agricultural big data, there is a need for efficient extraction, transformation and loading (ETL) workflows. These workflows specify the methods by which data coming from various sources will be inserted into the DWH after being appropriately processed. The following four studies aided in the selection of tools and methodologies for the quality management of real-time data.

In his study, Vassiliadis [10] provides a comprehensive analysis of ETL technologies, where he covers extraction challenges like keeping data quality during the ETL process, transformation methodologies, and data loading optimizations. Furthermore, he highlights the need to shift from batch processing to real-time ETL, supporting our use of Apache Kafka and Spark Streaming for real-time data ingestion.

Kakish and Kraft [11] mainly focus on real-time data warehousing and identify difficulties and solutions in continuous ETL processing. They propose event-driven ETL modifications, which are highly relevant to agricultural application, and require live data ingestion from IoT sensors or market feeds. In another study, Jain and Singh [12] further explore how to enforce quality assurance within the ETL process, and they propose validation mechanisms to maintain data consistency and accuracy.

Jia et al. [13] propose a real-time DWH architecture, where in their proposed setup they integrate log-based change data capture (CDC) and event-driven processing to minimize latency. Their work aligns with our approach for real-time agricultural data analytics and stresses the need for precomputed materialized views for efficient query performance.

2.3. Spatiotemporal and Agricultural Data Warehousing

Not all data are in the same format. Agricultural data originate from various sources and have different forms. Each data form needs to have its own specified channel and process to be inserted into the DWH, and the data warehouse needs to be set up in such a way that it can handle and utilize those data. The following three studies helped the development of the frameworks to handle spatiotemporal agricultural data within the proposed system.

The work of Wisnubhadra et al. [14] introduces a spatiotemporal data warehouse for agricultural analytics, and with this setting, they address the limitations in the case of spatial and temporal data integration. Their proposed setup can integrate open government data and geospatial analytics, supporting our argument that DWH systems must handle diverse agricultural datasets, including climate, soil, and market data.

Komasilovs et al. [15] design a cloud-based DWH for precision livestock farming (PLF), in which they integrate sensor-based livestock monitoring systems. Their approach highlights the need for automated ETL pipelines in farm management systems (FMISs), and in doing so, it reinforces the importance of modular ETL workflows in the proposed system.

In their study, Sayed et al. [16] analyze big data architectures for agriculture in developing countries. They find that architectures that combine both batch and real-time data processing are increasingly adopted, adding arguments on the need for flexible, scalable and versatile systems able to handle both data ingestion types.

2.4. Big Data and IoT in Agriculture

Some agricultural data are previously created and stored in databases in a specific form and others are created in real time. To make use of the valuable information contained in both, the DWH needs to be adequately equipped to handle them. The following five studies guided the approach to handling the import of real-time agricultural data and their analysis.

El Aissi et al. [17] introduce in their work a big data architecture for smart farming, where they combine data lakes and lambda architectures for real-time and batch data processing. Their work supports our use of Apache Kafka, Spark, and Hadoop for scalable agricultural analytics.

In their study, Jaiswal et al. [18] explore IoT-based smart agriculture solutions. The focus is on low-cost sensor integration for precision farming in developing nations. They perform real-time monitoring, cloud-based analytics, and automated decision-making, reinforcing the important role of the ability to support real-time agricultural data warehousing in our system.

McCarren et al. [19] develop an agri-data warehouse for predictive analytics. In their system, they use anomaly detection in data transformation workflows, with the goal being to improve the forecasting accuracy for commodity prices and yields. Their work supports our focus on the creation of automated ETL validation and having data quality assurance.

San Emeterio et al. [20] address the difficulties that derive from real-time agricultural data management. To achieve this, they propose a spatiotemporal semantic model for IoT-driven precision farming. Their work integrates time-series databases (InfluxDB) and semantic middleware (DAM&DQ), and in this way, it confirms the need for real-time agricultural decision-support systems. In a different study, San Emeterio et al. [21] further highlight the difficulties of processing the massive real-time data coming from precision agriculture.

Osinga et al. [22], in their work, link the potential of big data to practical farm-level benefits. They support the necessity of tailored scalable systems, reinforcing our proposed flexibility and modularity options.

The review of the related work summarized in Table 1 highlights the need for scalable, real-time, and spatiotemporal-able/aware DWH architectures in agriculture. The previous studies validate our use of the technologies and frameworks of HDFS, Hive, and Spark as a distributed processing framework, and at the same time, they point out the necessity of ETL optimizations, real-time data ingestion, and integration with the IoT and big data technologies. Our research builds upon these foundations, and we propose a flexible, extensible,

and secure agricultural DWH system that is capable of handling massive, heterogeneous datasets for performing predictive analytics and supporting decision-making.

Table 1. Summary of the related work and the relevance to this study.

Paper	Focus Area	Key Contribution	Technologies Discussed	Relevance to This Study
Yang, Ge and Helfert [5]	DWH Architecture	Identification of 9 representative architectures	DWH architecture	Supports architecture choice and use of scalable DWH frameworks
Chaudhuri and Dayal [6]	OLAP in DWH	Query optimization strategies for big data analytics	OLAP, MOLAP, ROLAP	Supports choice of Apache Hive for analytics
Jameel et al. [7]	ETL Optimization	Distributed vs. monolithic ETL	Spark, Flink, Kafka	Reinforces need for distributed ETL in agriculture
Thesma et al. [8]	Distributed Processing	Low-cost distributed computing for cotton phenotyping	Hadoop, distributed clusters	Supports use of scalable Hadoop-based systems for agricultural analytics
Cravero et al. Review [9]	Big Data Tools	Systematic review on Hadoop, Hive, Spark in agri-data	Hadoop, Hive, Spark	Validates Apache tools as standard for agri-big-data processing
Vassiliadis [10]	ETL Process	Survey of ETL techniques, real-time data integration	Apache Kafka, Spark Streaming	Validates real-time ingestion strategies
Kakish and Kraft [11]	Real-Time Data Integration	ETL for real-time data streaming	Apache Kafka, Spark Streaming	Supports real-time analytics and decision-making
Jain et al. [12]	Near Real-Time DWH	Hybrid refresh approach for real-time data updates	Change data capture (CDC), log-based ETL	Provides efficient data refresh model for agricultural analytics
Jia et al. [13]	Real-Time DWH	RTDW architecture with materialized view optimization	Change data capture (CDC), hybrid query execution	Reinforces event-driven ETL for real-time agricultural decision-making
Wisnubhadra et al. [14]	Spatiotemporal DWH	Open-source agricultural DWH with spatial analytics	SIDeKa, OLAP	Supports spatial integration in agricultural DWH
Komasilovs et al. [15]	Precision Livestock Farming	Cloud-based DWH integrating FMIS for small-scale farms	Cloud-based data warehouse (DW), data vaults	Highlights modular DWH architectures for livestock management
Sayed et al. [16]	Agri-Big-Data Architecture	Comparative study of batch + real-time architectures	Hybrid frameworks	Supports combined batch/stream architecture in agri-DWH systems
El Aissi et al. [17]	Big Data in Agriculture	Lambda architecture for batch and real-time processing	Data lakes, Kafka, Hadoop	Reinforces need for flexible data processing
Jaiswal et al. [18]	IoT in Smart Agriculture	Low-cost, scalable IoT framework for developing nations	Wireless sensor nodes (WSNs), cloud computing	Justifies IoT integration for data collection in agriculture

Table 1. *Cont.*

Paper	Focus Area	Key Contribution	Technologies Discussed	Relevance to This Study
McCarren et al. [19]	Predictive Analytics in Agriculture	Agri-data warehouse integrating real-time anomaly detection	Agri DWH, anomaly detection, ETL validation	Supports ETL-driven anomaly detection for agricultural forecasting
San Emeterio et al. [20]	Spatio-Temporal Data Management	Semantic middleware for agricultural data streams	InfluxDB, DAM&DQ, AFarCloud	Supports time-series optimization in agricultural IoT data
San Emeterio et al. [21]	IoT and Precision Agriculture	Spatio-temporal semantic model	AFarCloud, DEMETER	Justifies interoperability strategies in agricultural data integration
Osinga et al. [22]	Scalability in Agri-Data	Conceptual framework for big data readiness in farming	Big data systems, cloud	Justifies modular, farm-adapted scalable data systems

3. Materials and Methods

Based on the research that was performed and the needs of a state-of-the-art modern data warehouse (DWH), this section presents the design and implementation of our agricultural big data DWH following the structure in Figure 1.

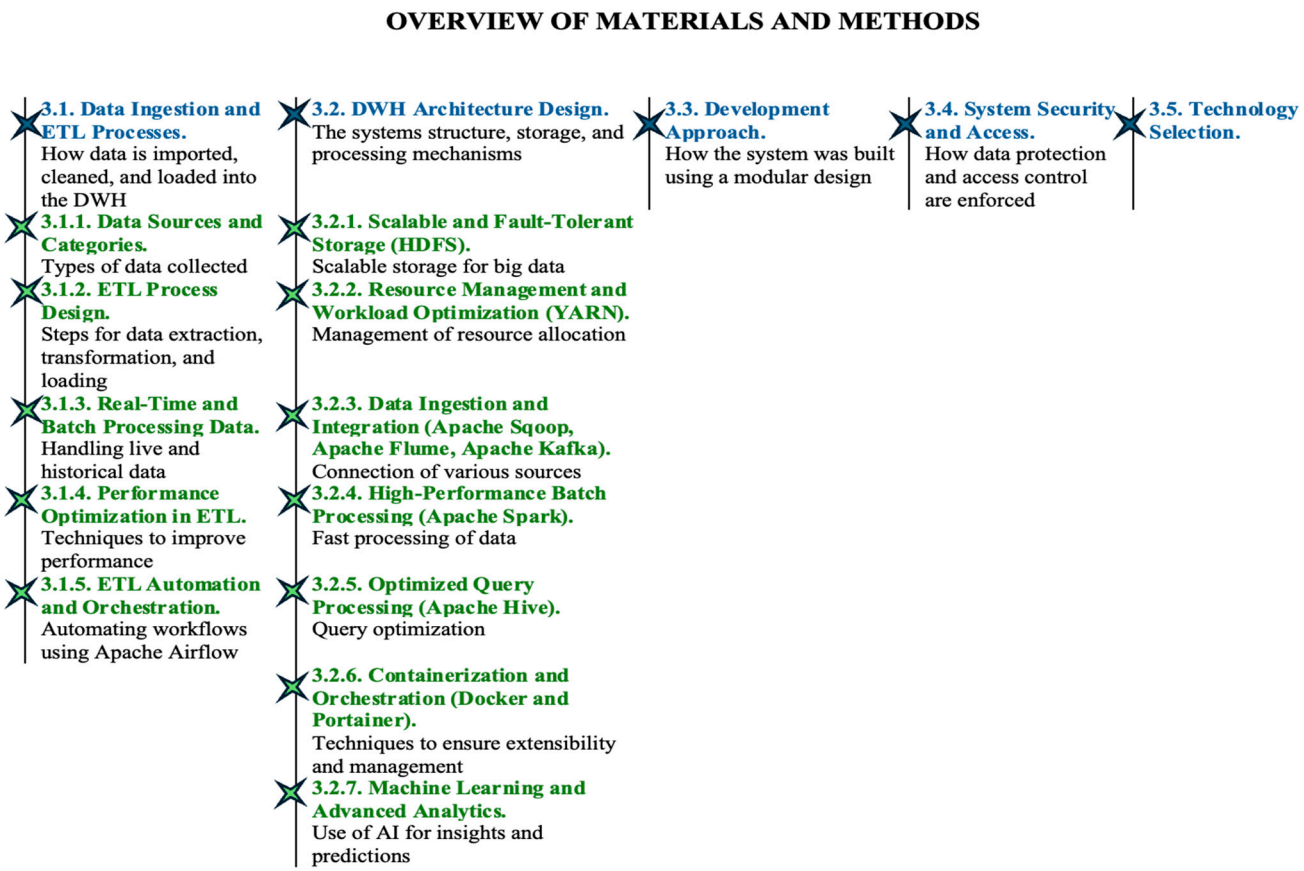

Figure 1. Section 3's structure.

The system can provide a scalable, efficient, and secure platform for data ingestion, processing, and analysis. With the integration of cutting-edge distributed computing technologies, the system can handle large-scale heterogeneous agricultural datasets. Some

of these can be real-time IoT sensor readings, satellite imagery, climate models, and market transactions. Table 2 presents all the tools and technologies that comprise the proposed agricultural big data DWH.

The following subsections provide a detailed breakdown of the system's components (Table 1), which execute the data-ingestion processes, the storage and query optimization strategies, and the scalability mechanisms; apply the security framework; and allow for predictive analytics models, which together create a high-performance, extensible, and robust agricultural data warehouse solution.

Table 2. Summary of all the used tools.

	Tool	Purpose
Data Ingestion and ETL Extract, transform, and load (ETL) data from various sources.	Apache Sqoop	Extracts structured data from relational databases and loads it into HDFS.
	Apache Flume	Ingests semi-structured and log-based data, commonly used for IoT logs and event data.
	Apache Kafka	Handles real-time streaming data ingestion from IoT sensors and external agricultural feeds.
	Apache Spark Streaming	Processes real-time data streams in parallel for real-time analytics.
	Apache Airflow	Manages workflow automation and scheduling for ETL processes.
Storage and Processing Store and process large-scale agricultural data.	Hadoop Distributed File System (HDFS)	Provides fault-tolerant, distributed storage for large datasets.
	Apache Spark	High-performance in-memory computing for batch processing, transformations, and machine learning.
	Apache Hive	SQL-like query processing for structured agricultural data stored in HDFS.
	Apache YARN	Manages and optimizes cluster resources, balancing workloads between Spark, Hive, and ingestion tasks.
Query and Analytics Querying, visualization, and monitoring of the DWH.	Apache Superset	Business intelligence tool for creating dashboards and interactive analytics.
	Grafana	Real-time monitoring and visualization of agricultural data.
	Precomputed Materialized Views	Optimizes query performance by storing frequently accessed aggregations.
	Indexing strategies	Optimizes query performance.
Security and Access Control Authentication, authorization, and compliance.	Role-Based Access Control (RBAC)	Implements fine-grained user authentication and data access permissions.
	Data Encryption	Protects sensitive agricultural data.
Containerization and Orchestration Deploy, scale, and manage the entire DWH system.	Docker	Containerizes all components, ensuring portability and modularity.
	Portainer	Provides a lightweight web-based UI for managing Docker containers.

Table 2. *Cont.*

	Tool	Purpose
Supporting Databases Store metadata and manage schema definitions.	PostgreSQL	Stores metadata for Apache Hive Metastore and external transactional data.
	Apache Hive Metastore	Manages metadata for Hive and Spark tables.
Machine Learning and Advanced Analytics Predictive analytics, anomaly detection, and AI-driven insights.	Apache Spark MLlib	Runs machine learning models (e.g., ARIMA, LSTMs) for yield forecasting and price predictions.
	Python 3.10, ML Stack (Scikit-Learn, TensorFlow, PyTorch 2.1)	Supports AI-driven insights, anomaly detection, and predictive modeling.

3.1. Data Ingestion and ETL Processes

The first and critical step is efficient data ingestion and transformation. The proposed system applies a hybrid extract, transform, load (ETL) process to handle structured, semi-structured, and unstructured data. It also integrates both batch processing for historical data and real-time streaming for dynamic datasets. The ETL framework makes sure that there is consistency, accuracy, and usability of data by leveraging Apache Sqoop, Apache Flume, Apache Kafka, Apache Spark Streaming, and Apache Hive.

3.1.1. Data Sources and Categories

Agricultural data can originate from a vastly diverse range of sources. Each of these sources requires specific specialized ingestion techniques. Our system categorizes the data sources in three ways. First there are structured data, which relational databases (e.g., PostgreSQL, MySQL) carry, containing market transactions, historical crop yields, policy data, etc. Then, there are semi-structured data, like CSV/JSON/XML-based meteorological reports, governmental agricultural datasets, and supply chain data. A third variation is unstructured data, as from IoT sensor readings (soil moisture, temperature), satellite imagery, and drone-captured field data. Each type of data requires its own optimized extraction method to ensure that the data are efficiently stored and retrieved.

3.1.2. ETL Process Design

The hybrid ETL pipeline consists of three key stages. The first stage is data extraction, in which data are retrieved from various sources using specialized tools. Apache Sqoop extracts structured data from relational databases and transfers it to HDFS. Apache Flume ingests semi-structured and log-based data streams, commonly used for IoT logs and event-based data. Apache Kafka and Spark-Streaming facilitate real-time ingestion, streaming high-velocity data from IoT sensors and external agricultural feeds.

The second stage is data transformation, in which raw data undergo preprocessing, validation, and enrichment. Apache Spark performs distributed in-memory transformations to clean, normalize, and aggregate data. Feature extraction techniques are applied to sensor and climate data to ensure compatibility with analytical models. Data quality checks (deduplication, anomaly detection) are embedded in Spark's transformation phase.

The third stage is data loading, in which the processed data are stored and made available for querying and analysis. Batch data that have been processed are loaded into Apache Hive, enabling OLAP-style queries for trend analysis and decision-making. Real-time data like sensor-based streaming data are continuously updated in pre-aggregated tables using Kafka-to-Hive integration.

Real-time ingestion techniques use Kafka and Spark to handle the data that arrive late, to help maintain consistency. Kafka retains data for a preset duration, allowing Spark to reprocess delayed records. Using a "watermarking" mechanism, Spark ensures that the late-arriving data will be correctly assigned to the original time interval.

3.1.3. Real-Time and Batch Processing Data

The system makes a distinction in the handling of batch and real-time data to optimize performance. Batch processing is suitable for large-scale historical datasets like market trends, and policy data updates. Sqoop and Spark are being utilized to handle batch processing efficiently. Real-time streaming data are the continuously incoming data provided by sensors, real-time pricing updates, and climate monitoring. Kafka and Spark Streaming process and analyze these real-time data streams in near real time and provide decision-makers with up-to-date insights.

3.1.4. Performance Optimization in ETL

Using Hive, large datasets can be partitioned and indexed. Hive partitions these large datasets based on various criteria (by time, region, and data type), and this greatly reduces the query response times. Furthermore, by caching frequently accessed queries, redundant computations are minimized, and this enhances performance (materialized views are precomputed). Finally, Spark-based machine learning models monitor and detect outliers, anomalies, and inconsistencies in incoming data streams.

3.1.5. ETL Automation and Orchestration

To streamline the ETL workflows and avoid manual intervention as much as possible, the proposed system uses Apache Airflow, which automates and performs the scheduled ETL jobs and ensures the systematic data ingestion. Also, Dockerized deployment makes the DWH scalable. All the ETL components run in Docker containers, managed and monitored through Portainer. This makes the setting easy to expand as additional needs arise. Moreover, Grafana and Superset provide dashboards for ETL monitoring and logging of performance enabling troubleshooting. This robust ETL pipeline warrants that the agricultural data are consistently ingested, transformed, and made available for analytics. Thus, our system supports stakeholders in decision-making across the agricultural supply chain.

3.2. Data Warehouse (DWH) Architecture Design

The proposed data warehouse (DWH) is designed to be scalable, fault tolerant, high performing, and able to manage recourses efficiently. It also integrates data seamlessly with the integration of batch data and real-time data. It provides optimal query performance, and it enables advanced analytics. The system follows a modular, containerized approach.

The architecture is structured around the following six core functional components.

3.2.1. Scalable and Fault-Tolerant Storage (HDFS)

As the basis of the DWH's storage layer there is the Hadoop Distributed File System (HDFS). HDFS is optimized for big data storage and provides fault tolerance. In HDFS, the data are automatically replicated in multiple nodes to prevent data loss in case of failures (Figure 2). This process of replication is what makes the DWH fault tolerant. Additionally, the ability to add new nodes as needed to accommodate increasing data volumes, which makes the system highly scalable. HDFS is designed for storage and retrieval of datasets that are very large. Datasets like climate models, IoT sensor logs, and satellite imagery. To increase flexibility, in the DWH setting proposed, HDFS is deployed within Dockerized containers. This allows for the storage management to be easily be scaled across different environments.

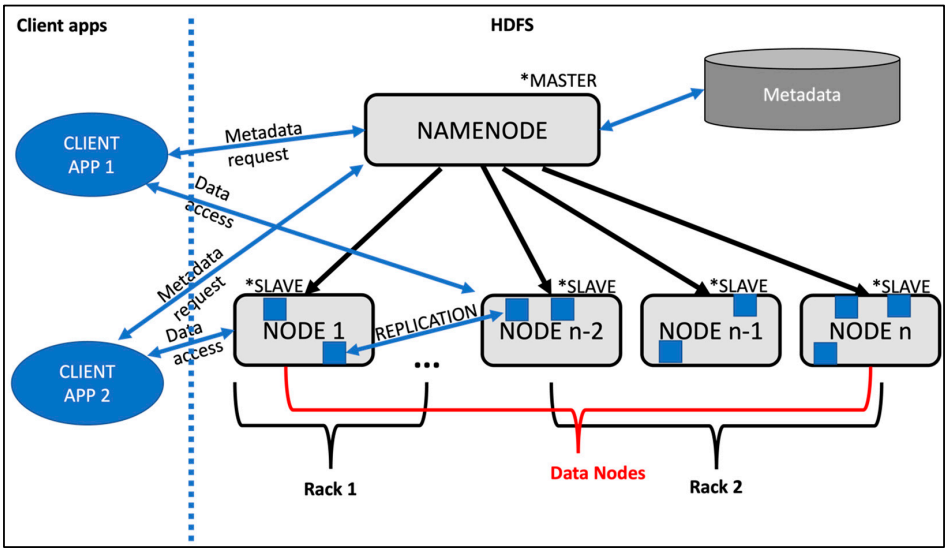

Figure 2. Graphical representation of HDFS (This figure is an original illustration created by the authors to depict the structure of HDFS as applied in our system. While the design is inspired by standard architecture diagrams from Apache documentation, we have adapted it to highlight the components relevant to our implementation. The figure does not affect scientific understanding and aims to enhance clarity).

3.2.2. Resource Management and Workload Optimization (YARN)

High-performance processing is based in efficient resource allocation. The DWH system uses Yet Another Resource Negotiator (YARN) to manage and distribute computing resources, and it does so dynamically (Figure 3). With YARN, the system can have the optimal CPU and memory utilization for all workloads; it can support concurrent data ingestion, processing, and analytics tasks; and it can efficiently balance both batch and real-time workloads with low-latency query execution. YARN integrates with Apache Spark and Hive and ensures that distributed processing tasks are executed without resource bottlenecks.

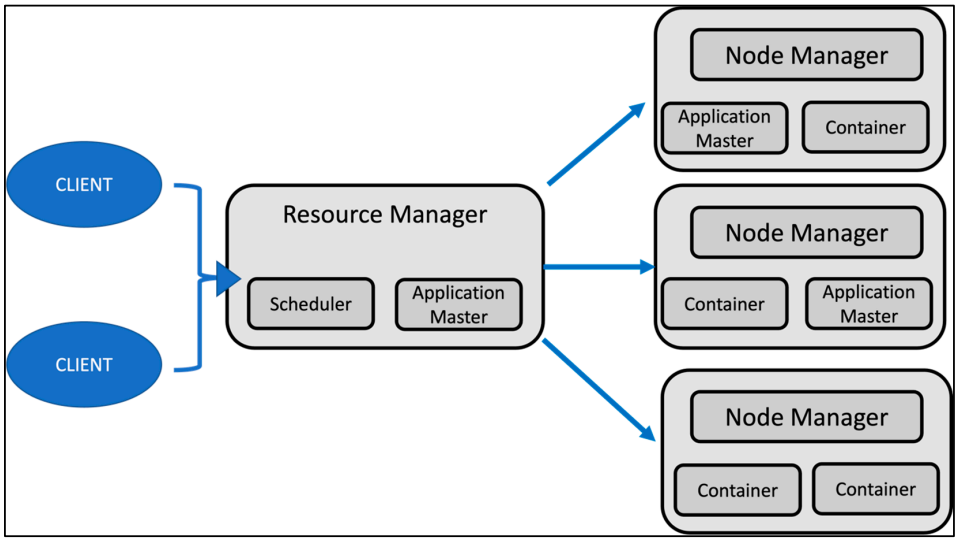

Figure 3. Graphical representation of YARN (This figure is an original illustration created by the authors to depict the structure of HDFS as applied in our system. While the design is inspired by standard architecture diagrams from Apache documentation, we have adapted and simplified it to highlight the components relevant to our implementation. The figure does not affect scientific understanding and aims to enhance clarity).

3.2.3. Data Ingestion and Integration (Apache Sqoop, Apache Flume, Apache Kafka)

Agricultural datasets are, by their very nature, diverse, coming from various heterogeneous sources. To handle such data, the system applies a hybrid ingestion model. This approach allows the DWH to take data both in batches (historical data) and from real-time streaming sources (continuously generated datasets).

To ingest batch data, the system uses Apache Sqoop. With Sqoop, structured data can be extracted from relational databases (e.g., PostgreSQL, MySQL). For large agricultural records (e.g., market prices, government policy data), the input process can be parallelized to facilitate faster import times. Additionally, Sqoop reduces data duplication with the support of incremental loading.

For real-time streaming data, the system uses Apache Flume and Apache Kafka. Flume is used to import semi-structured data that originate from IoT sensor logs or satellite image metadata. Kafka is used to manage high-velocity streaming data, supporting event-driven triggered data creation. Spark Streaming is used to process the Kafka streams, allowing for real-time analytics on weather patterns, soil conditions, and market fluctuations.

Since a part of agricultural data are of a spatiotemporal nature, the proposed system is designed to efficiently handle them. Apache Hive can partition time-series data and Apache Spark can perform distributed spatial processing. This setting can also support geospatial analytics for climate and soil data. Agricultural decision-makers, from policymakers to agro-commodity traders, can analyze data with both temporal and spatial dimensions for more informed insights.

With this setting, the system offers seamless integration of batch and real-time data and gives users that are decision-makers the ability to access both historical trends and live insights.

3.2.4. High-Performance Batch Processing (Apache Spark)

Apache Spark is the basic ETL engine. Spark can process distributed data and perform batch and real-time transformations. Spark processes data 10–100$\times$ faster than the traditional MapReduce through the ability to perform in-memory processing. It also handles the cleaning, normalization and feature engineering of agricultural datasets. Spark additionally supports predictive analytics with the integration of machine learning algorithms. This can be used by decision-makers for price forecasting, yield prediction, and climate impact analysis. Additionally, Spark can run across distributed clusters, which is what gives scalability to the system.

3.2.5. Optimized Query Processing (Apache Hive)

Apache Hive has a query interface connected to the DWH, and it allows users to execute analytical queries on large datasets. Hive uses an SQL-like query language (HQL), enabling users to write queries using familiar SQL syntax. It optimizes query performance by partitioning and indexing. This is the process of segmenting data based on various selected criteria, which greatly enhances performance. Hive allows the use of precomputed materialized views. This is the ability to store frequently accessed aggregations to reduce the query execution time. Lastly, Hive enables distributed query execution, with Spark's in-memory processing, producing faster results.

3.2.6. Containerization and Orchestration (Docker and Portainer)

The system is deployed in containers, using Docker and managed with Portainer, to provide modularity, ease of deployment, and scalability. Each container has one or more of the software components of the DWH in a separate Linux environment. The Dockerized deployment ensures that more nodes can be added across cloud and on-premises setups,

allowing for more computing power and storage (horizontal scaling). The use of Portainer for management provides a web-based interface for monitoring and handling these Docker containers. Having each software component in a separate container makes it easy to update, restart, or replace them without totally disrupting system operations.

3.2.7. Machine Learning and Advanced Analytics

Finally, the proposed system can perform advanced predictive analytics, detect anomalies, and offer decision-support mechanisms. This is achieved by utilizing machine learning (ML) models (for forecasting agricultural trends, price movements, and climate risks) from Apache Spark's MLlib or Pythons ML stack (Scikit-Learn, TensorFlow, PyTorch) and various other common ML models (ARIMA, LSTMs, etc.). Anomaly detection techniques are embedded through Apache Kafka in the data input phase and through Hive with periodic batch anomaly detection running on historical data within the ETL pipeline to identify inconsistencies and ensure data integrity. Grafana and Apache Superset allow for access to visual dashboards that provide farmers, policymakers, and agribusiness stakeholders with actionable insights derived from the vast amounts of agricultural data collected.

To avoid slowing down the system, the ML models can be trained in batch mode and then deployed only after optimization. The models can be scheduled to run separately from core data operations, triggered either periodically or based on data ingestion. This plan ensures that predictive analytics are included without affecting the system's performance (particularly during highly resource-demanding operations).

3.3. Development Approach

A modular and iterative development approach has been followed to ensure system flexibility and extensibility, while at the same time allowing for step-by-step verification that each phase's components are fully functional. For the proposed DWH, the model of continuous integration and deployment (CI/CD) has been followed. With Docker-based deployment, every component (HDFS, Spark, Hive) runs in containerized environments, making scaling an easy process. The development is iterative to align with evolving agricultural data needs. Using this methodology, the system's functionality can continually be verified, and the DWH is kept in sync with the ongoing requirements. To achieve this, the DWH setting has been separated into 3 distinct phases. In Phase 1, we complete the initial DWH setup with the core HDFS, Hive, and Spark. In Phase 2, we perform the integration of Kafka and Flume for data ingestion. In Phase 3, we perform the optimization via query indexing, caching, and implementation of machine learning pipelines.

3.4. System Security and Data Access Control

Security and data protection are two major components of the proposed DWH system. To make the DWH secure and able to handle sensitive personal data, strict access control is run so the system adheres to data protection regulations. Passwords are required for all user accounts, and users will be denied or granted access to data based upon their assigned role. This guarantees that only permitted users can access sensitive data. The DWH will also make use of role-based access control (RBAC), in which there are access levels that will be set for various users. This helps prevent unauthorized access to critical data and ensures that each user can only access the data relevant to their role within the system. With the addition of storage encryption, we ensure compliance with data protection regulations, and we safeguard sensitive agricultural datasets. In more detail, RBAC is implemented throughout the data pipeline by assigning permissions to predefined roles. User roles, such as analysts, admins, engineers, and external auditors, are granted access only to the data and processes that are relevant to the role's responsibilities, ensuring that sensitive information is protected. Data encryption is applied both at rest and in

transit. The HDFS blocks are encrypted using AES-256 with Hadoop's encryption feature. The communication between services (Kafka, Spark, Hive) is secured utilizing SSL/TLS protocols. These measures support compliance with data protection frameworks, such as the GDPR.

3.5. Technology Selection

During the design phase of the proposed system, our choices were guided by attempts to ensure great performance, integration flexibility, and large community support. Apache Hive was selected over Presto due to its stronger compatibility with batch-oriented ETL processes and greater integration with the HDFS protocol. Spark was chosen over Jlink for the widely used and mature machine learning library MLlib integration and the great community documentation and support. Docker was chosen over the powerful orchestration offerings of Kubernetes for the prototype phase for the simplicity and faster deployment that a small to medium cluster environment needs. These selections make for a foundation that is both flexible and scalable and can modularly be replaced as future needs arise.

4. Results

The implementation of the proposed agricultural big data warehouse manages to successfully integrate all the parts that are necessary for a robust and scalable system that is able to handle massive, diverse datasets and enable users to perform data analytics. Figure 4 graphically presents the resulting setup from the application of the tools and software that have been discussed in Section 3.

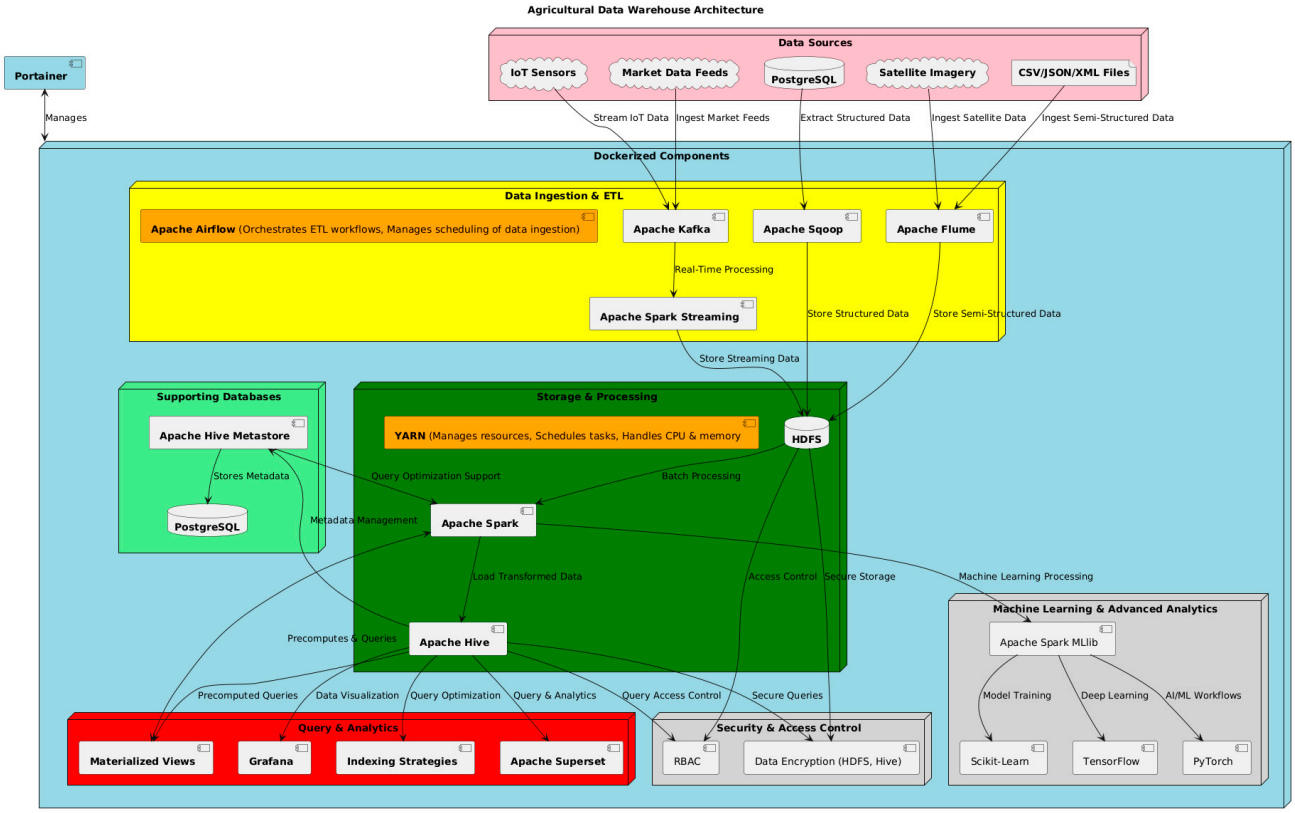

Figure 4. Graphical representation of the proposed agricultural data warehouse architecture (authors' own work).

The use of Apache Sqoop, Flume, Kafka, Spark-Streaming, and Airflow in the data ingestion and ETL phase make the extraction, ingestion, scheduling, handling and processing of data highly efficient and capable of managing highly varied data source inputs.

HDFS, YARN, Hive, and Spark, used for storage and processing, lead to the high-performance, fault-tolerant storage of the agricultural data.

Apache Superset, alongside Grafana, with the additional implementation of precomputed materialized views and indexing strategies, provides the end user with real-time monitoring and optimized performance in their effort to obtain interactive analytical results.

The enforcement of user authentication through role-based access control, and data encryption, keeps the sensitive agricultural datasets protected.

The containerized deployment through Docker ensures portability and modularity, while the use of Portainer allows for uncomplicated management of the Docker containers. Also, the use of Docker in combination with HDFS allows for growth of the DWH system based on the data needs.

The integration of Apache Spark MLlib and Python ML Stack supports AI-driven insights and application of state-of-the-art predictive and analytics algorithms.

Finally, the iterative development approach that was followed ensures the ability to continuous refine all the DWH components.

4.1. Query Performance Benchmarks

A series of benchmarking experiments were conducted on the proposed system, using demo datasets that had been created. The data simulated typical agricultural parameters, and they included timestamps, locations, temperature, soil moisture, and yield estimates. The manufactured datasets contained 100,000, 1 million, and 10 million rows. Using Python3, Pandas, and Numpy, sensor data were created with the following script.

```
import pandas as pd
import numpy as np

def generate_csv(rows, filename):
df = pd.DataFrame({
'ts': pd.to_datetime(np.random.randint(1672531200, 1704067200, size=rows), unit='s'),
'location': np.random.choice(['Field_A', 'Field_B', 'Field_C'], rows),
'soil_moisture': np.random.uniform(10, 40, rows),
'temperature': np.random.uniform(15, 35, rows),
'yield_estimate': np.random.randint(200, 500, rows)
})

df.to_csv(filename, index=False)
print(f"{filename} created with {rows:,} rows.")

generate_csv(100000, "agri_data_100K.csv")
generate_csv(1000000, "agri_data_1M.csv")
generate_csv(10000000, "agri_data_10M.csv")
```

A set of representative queries, including the full row counts, group-by aggregations, filtered aggregations, and top-k retrievals, provided the execution times used to record the scalability behavior of the system. Table 3 presents the results and the queries used for the three different dataset sizes. The system handled the most demanding queries in under 45 s and the less demanding ones in under 2 s.

In addition to the query performance, the ingestion time was also evaluated to assess how the system responds to data uploads. The three CSV files of increasing size were ingested in the data warehouse pipeline by executing the "LOAD DATA LOCAL INPATH" command via Hive. The 100,000-row file was ingested in 0.577 s, the 1-million-row file in

0.847 s, and the 10-million-row file in 3.162 s. The system displayed the ability to handle growing data volumes with minimal delay.

Table 3. Execution performance summary.

Query Description	SQL Query	Execution Time (s)	Dataset Size	Purpose
Total Row Count	SELECT COUNT(*) FROM agri_data_100K;	1.675	100 K	Validates ingestion and basic query scalability.
Total Row Count	SELECT COUNT(*) FROM agri_data_1M;	1.629	1 M	Validates ingestion and basic query scalability.
Total Row Count	SELECT COUNT(*) FROM agri_data_10M;	7.634	10 M	Validates ingestion and basic query scalability.
Average Yield Estimate per Location	SELECT location, AVG(yield_estimate) FROM agri_data_100K GROUP BY location;	1.738	100 K	Tests group-by aggregation on a medium dataset.
Average Yield Estimate per Location	SELECT location, AVG(yield_estimate) FROM agri_data_1M GROUP BY location;	2.635	1 M	Tests group-by aggregation on a large dataset.
Average Yield Estimate per Location	SELECT location, AVG(yield_estimate) FROM agri_data_10M GROUP BY location;	15.653	10 M	Tests group-by aggregation at high scale.
Average Soil Moisture with Condition (temp > 25)	SELECT location, AVG(soil_moisture) FROM agri_data_100K WHERE temperature > 25 GROUP BY location;	1.742	100 K	Tests conditional filtering + aggregation on moderate data.
Average Soil Moisture with Condition (temp > 25)	SELECT location, AVG(soil_moisture) FROM agri_data_1M WHERE temperature > 25 GROUP BY location;	3.647	1 M	Tests conditional filtering + aggregation on large data.
Average Soil Moisture with Condition (temp > 25)	SELECT location, AVG(soil_moisture) FROM agri_data_10M WHERE temperature > 25 GROUP BY location;	29.726	10 M	Tests complex filtering and aggregation at high scale.
Top 10 Rows by Yield Estimate	SELECT * FROM agri_data_100K ORDER BY yield_estimate DESC LIMIT 10;	1.645	100 K	Validates performance of sorting and top-k selection.
Top 10 Rows by Yield Estimate	SELECT * FROM agri_data_1M ORDER BY yield_estimate DESC LIMIT 10;	5.598	1 M	Validates performance of sorting and top-k selection.
Top 10 Rows by Yield Estimate	SELECT * FROM agri_data_10M ORDER BY yield_estimate DESC LIMIT 10;	44.724	10 M	Validates performance of sorting and top-k selection at large scale.

4.2. Use Case Illustration: Yield Estimation Query

As a demonstration of the possible practical application of the system, we constructed a use case where the average yield estimation per field was simulated. Using Hive, we executed:

SELECT location, AVG (yield_estimate) FROM agri_data_1M GROUP BY location;

This query processed over 1 million records in approximately 2.6 s, showcasing the value of the system's analytical capabilities for decision-makers who need real-time insights into field productivity.

4.3. Fault Tolerance Tests

The system was tested for fault tolerance by conducting the following fault injection experiments on the key components of the data pipeline. While a long-running aggregation query was active, the hive server was forcefully stopped and restarted. As expected, the query failed without corrupting the metadata or impacting other services. Upon restart, all the following queries ran normally, confirming Hive's ability to recover with minimal to no disruption. To test HDFS, a DataNode was terminated during an active query. With the HDFS replication set to the default three nodes, the system retrieved data blocks from replica nodes and the query completed successfully, demonstrating HDFS's tolerance to node failures. Lastly, after forced shutdowns and restarting the entire Docker cluster, all the previously ingested and queried datasets, alongside all the table structures, remained available, verifying that persistent volumes and metadata were preserved.

5. Discussion

The implementation of a data warehouse (DWH) is the only available option for when datasets are massive. Traditional relational database management systems (RDBMSs) have limited abilities for scaling and handling unstructured data. They are primarily designed for transactional processing (OLTP) and struggle with large-scale analytical workloads (OLAP) because they do not apply distributed storage and parallel processing in the same way a DWH functions. Additionally, they are not optimized for semi-structured and unstructured data from sources such as IoT devices or satellite imagery. Although conventional databases excel in handling structured transactional data, DWHs are designed for large-scale analytical processing, and they could integrate diverse data sources through optimized extract, transform, load (ETL) workflows, futureproofing in terms of further data collection and storing. The heterogeneity of agricultural data, which originate from diverse sources such as IoT sensors, satellite imagery, and market transactions, highlights the necessity of a DWH solution in the agricultural domain.

The increasing volume, velocity, and variety of data call for modern architectures such as the proposed data warehouse (DWH) to ensure efficient data management and utilization. The traditional storage and processing infrastructures are inadequate to meet the growing data demands. Therefore, organizations must invest in scalable, high-performance solutions to fully leverage big data for decision-making and operational efficiency [23]. There is a great need for robust big data analytics capabilities (BDACs) to manage and extract valuable insights from these vast datasets, and in this way, to ensure that organizations move forward in the way of sustainable innovation and organizational competitiveness [24]. Furthermore, as industries increasingly rely on big data for decision-making, the ability to integrate and process these vast information streams is becoming a key determinant of innovation performance [25].

However, despite the unquestionable benefits of a DWH, the decision to implement one must be carefully evaluated by weighing the complexity of the system and the actual data requirements. The distributed storage, the multi-stage query execution, and the container communication cause an overhead that means that a DWH may not always be the optimal choice for smaller datasets. When dealing with moderate-sized datasets, which are in the order of millions rather than petabytes of records, an out-of-the-box relational database may provide faster query performance with a significantly lower setup

configuration. The modular and containerized architecture of a DWH introduces inherent latency, as multiple processes need to interact across distributed components to execute a single query.

Furthermore, to implement a high-performance DWH, it is required that experts in a diverse set of domains, like distributed computing, container orchestration, security frameworks, and real-time processing technologies, work together to achieve it. Without careful planning, the system may face inefficiencies, leading to suboptimal performance even with the use of state-of-the-art tools. Therefore, organizations must assess whether their data needs justify the required effort and use of resources of a DWH before proceeding with its deployment.

Despite these challenges, the proposed DWH architecture provides a scalable and efficient solution for managing agricultural big data. By leveraging the power of Apache Spark, Hive, and HDFS, alongside the other proposed software components, the system ensures efficient query execution, real-time analytics, and fault tolerance.

To accommodate smaller farms and stakeholders with limited technical resources, the system was designed to be modular and Docker-based. A minimal deployment can be set up on a single-node server by a skilled system administrator relatively fast. In this simplified version, the overall cost of the deployment and maintenance of the system would be the administrator's wage and the PC that would be used as a server for the system, which would be different based on several factors, like the country, the administrator's skills, etc. Acknowledging that this can still be a cost that a small farm cannot manage, the system can be shared by more than one entity, with separate access and rights for each. This way, the cost would be also shared. The system has the flexibility to scale down to local deployments or scale up to region-level application.

Our proposed system was benchmarked with datasets of up to 10 million records, but full-scale stress tests with high frequency real-time data streams ingestion remain a topic for future work. Although the architecture is designed to scale by adding nodes and computing resources, we recognize that bottlenecks may arise in the disk I/O, memory, or network saturation. In future deployments, we plan to stress test the system under realistic streaming workloads to quantify the performance barriers.

Future optimizations may focus on improving the communication efficiency between containers and refining the indexing strategies to further reduce the query latency, making sure that the DWH continues to meet the ever-evolving data demands in agricultural research and decision-making.

Author Contributions: Software, A.T.; validation, K.M. and S.A.N.; writing—original draft preparation, A.T.; writing—review and editing, S.A.N., M.T. and S.R.-P.; supervision, K.M. and S.A.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union under the Horizon Europe H2020 grant for the BIOVALUE project, grant number 101000499, "Fork-to-farm agent-based simulation tool augmenting BIOdiversity in the agri-food VALUE chain". This work does not necessarily reflect the view of the EU and in no way anticipates the Commission's future policy.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Acknowledgments: We would like to thank Konstantinos Theofilou for his invaluable help and guidance in the implementation and design of the proposed agricultural big data DWH solution.

Conflicts of Interest: Author S.R.-P. was employed by the company IDENER. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DWH	Data Warehouse
RDBMS	Relational Database Management System
HDFS	Hadoop Distributed File System
DBMS	Database Management System
OLTP	Online Transactional Process
OLAP	Online Analytical Process
ROLAP	Relational Online Analytical Process
MOLAP	Multidimensional Online Analytical Process
HOLAP	Hybrid Online Analytical Process
ETL	Extraction Transformation Loading
RBAC	Role-Based Access Control
CDC	Change Data Capture
IoT	Internet of Things
PLF	Precision Livestock Farming
FMIS	Farm Management System
RTDW	Real-Time Data Warehouse
WSNs	Wireless Sensor Nodes
UI	User Interface
SQL	Sequel Query Language
ML	Machine Learning
AI	Artificial Intelligence
YARN	Yet Another Resource Negotiator
CPU	Central Processing Unit
HQL	Hive Query Language
CI/CD	Continuous Integration and Continuous Deployment
BDAC	Big Data Analytics Capabilities

References

1. Agarwal, I.; Rana, D.; Shah, P.; Dude, A.; Patel, P. Optimizing Crop Monitoring: A Data Warehouse Approach in Precision Agriculture. In Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 24–28 June 2024; pp. 1–7. [CrossRef]
2. Ngo, V.M.; Duong, T.V.T.; Nguyen, T.B.T.; Dang, C.N.; Conlan, O. A big data smart agricultural system: Recommending optimum fertilizers for crops. *Int. J. Inf. Tecnol.* **2023**, *15*, 249–265. [CrossRef]
3. Rana, H.; Farooq, M.U.; Kazi, A.K.; Baig, M.A.; Akhtar, M.A. Prediction of Agricultural Commodity Prices using Big Data Framework. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 12652–12658. [CrossRef]
4. Mutia, I.; Sitanggang, I.S.; Annisa, A.; Astuti, D.A. Application of Spatial Data Warehouse for Agriculture: Challenge and Future Trends. In Proceedings of the 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), Depok, Indonesia, 14–15 September 2021; pp. 277–282. [CrossRef]
5. Yang, Q.; Ge, M.; Helfert, M. Analysis of Data Warehouse Architectures: Modeling and Classification. In Proceedings of the 21st International Conference on Enterprise Information Systems, Heraklion, Greece, 3–5 May 2019; Science and Technology Publications, Lda: Setúbal, Portugal, 2019. [CrossRef]
6. Dayal, U.; Chaudhuri, S. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec. (ACM Spec. Interes. Gr. Manag. Data)* **1997**, *26*, 65–74. [CrossRef]
7. Jameel, K.; Adil, A.; Bahjat, M. Analyses the performance of data warehouse architecture types. *J. Soft Comput. Data Min.* **2022**, *3*, 45–57, Penerbit UTHM: Batu Pahat, Malaysia. [CrossRef]
8. Thesma, V.; Rains, G.C.; Mohammadpour Velni, J. Development of a Low-Cost Distributed Computing Pipeline for High-Throughput Cotton Phenotyping. *Sensors* **2024**, *24*, 970. [CrossRef] [PubMed]

9. Cravero, A.; Pardo, S.; Sepúlveda, S.; Muñoz, L. Challenges to Use Machine Learning in Agricultural Big Data: A Systematic Literature Review. *Agronomy* **2022**, *12*, 748. [CrossRef]
10. Vassiliadis, P. A survey of extract–transform–load technology. *Int. J. Data Warehous. Min. (IJDWM)* **2009**, *5*, 1–27. [CrossRef]
11. Kakish, K.; Kraft, T.A. ETL evolution for real-time data warehousing. In Proceedings of the Conference on Information Systems Applied Research ISSN, New Orleans, LA, USA, 26–30 March 2012; p. 1508. Available online: https://www.researchgate.net/profile/Theresa-Kraft/publication/280837435_ETL_Evolution_for_Real-Time_Data_Warehousing/links/56008f5308ae07629e52af09/ETL-Evolution-for-Real-Time-Data-Warehousing.pdf (accessed on 4 March 2025).
12. Jain, T.; Rajasree, S.; Saluja, S. Refreshing datawarehouse in near real-time. *Int. J. Comput. Appl.* **2012**, *46*, 24–29.
13. Jia, R.; Xu, S.; Peng, C. Research on Real Time Data Warehouse Architecture. In Proceedings of the International Conference on Information Computing and Applications, Singapore, 16–18 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 333–342. [CrossRef]
14. Wisnubhadra, I.; Kamal Baharin, S.S.; Herman, N.S. Open Spatiotemporal Data Warehouse for Agriculture Production Analytics. *Int. J. Intell. Eng. Syst.* **2020**, *13*, 419–431. [CrossRef]
15. Komasilovs, V.; Kviesis, A.; Zacepins, A.; Bumanis, N. Development of the data warehouse architecture for processing and analysis of the raw pig production data. *AGROFOR Int. J.* **2018**, *3*, 64–71. [CrossRef]
16. Sayed, S.A.; Mahmoud, A.S.; Farg, E.; Mohamed, A.M.; Saleh, A.M.; AbdelRahman, M.A.E.; Moustafa, M.; AbdelSalam, H.M.; Arafat, S.M. A Comparative Study of Big Data Use in Egyptian Agriculture. *J. Electr. Syst. Inf. Technol.* **2023**, *10*, 21. [CrossRef]
17. El Aissi, M.E.M.; Benjelloun, S.; Lakhrissi, Y.; Ali, S.E.H.B. A Scalable Smart Farming Big Data Platform for Real-Time and Batch Processing Based on Lambda Architecture. *J. Syst. Manag. Sci.* **2023**, *13*, 17–30. [CrossRef]
18. Jaiswal, S.P.; Bhadoria, V.S.; Agrawal, A.; Ahuja, H. Iternet of Things (IoT) for smart agriculture and farming in developing nations. *Int. J. Sci. Technol. Res.* **2019**, *8*, 1049–1056.
19. McCarren, A.; McCarthy, S.; Sullivan, C.O.; Roantree, M. Anomaly detection in agri warehouse construction. In Proceedings of the Australasian Computer Science Week Multiconference, January 2017, Geelong, Australia, 31 January–3 February 2017; pp. 1–10. [CrossRef]
20. San Emeterio de la Parte, M.; Martínez-Ortega, J.F.; Hernández Díaz, V.; Martínez, N.L. Big Data and precision agriculture: A novel spatio-temporal semantic IoT data management framework for improved interoperability. *J. Big Data* **2023**, *10*, 52. [CrossRef]
21. San Emeterio de la Parte, M.; Lana Serrano, S.; Muriel Elduayen, M.; Martínez-Ortega, J.-F. Spatio-Temporal Semantic Data Model for Precision Agriculture IoT Networks. *Agriculture* **2023**, *13*, 360. [CrossRef]
22. Osinga, S.A.; Paudel, D.; Mouzakitis, S.A.; Athanasiadis, I.N. Big Data in Agriculture: Between Opportunity and Solution. *Agric. Syst.* **2022**, *195*, 103298. [CrossRef]
23. Ghaleb, E.A.A.; Dominic, P.D.D.; Singh, N.S.S.; Naji, G.M.A. Assessing the Big Data Adoption Readiness Role in Healthcare between Technology Impact Factors and Intention to Adopt Big Data. *Sustainability* **2023**, *15*, 11521. [CrossRef]
24. Hao, S.; Zhang, H.; Song, M. Big Data, Big Data Analytics Capability, and Sustainable Innovation Performance. *Sustainability* **2019**, *11*, 7145. [CrossRef]
25. Alaskar, T.H.; Alsadi, A.K.; Aloulou, W.J.; Ayadi, F.M. Big Data Analytics, Strategic Capabilities, and Innovation Performance: Mediation Approach of Organizational Ambidexterity. *Sustainability* **2024**, *16*, 5111. [CrossRef]