scientific reports

Check for updates

OPEN

A user-friendly NoSQL framework for managing agricultural field trial data

Steven H. Wu[1]✉ & Tristan A. Mueller[2]

Field trials are one of the essential stages in agricultural product development, enabling the validation of products in real-world environments rather than controlled laboratory or greenhouse settings. With the advancement in technologies, field trials often collect a large amount of information with diverse data types from various sources. Managing and organizing extensive datasets can impose challenges for small research teams, especially with constantly evolving data collection processes with multiple collaborators and introducing new data types between studies. A practical database needs to be able to incorporate all these changes seamlessly. We present DynamoField, a flexible database framework for collecting and analyzing field trial data. The backend database for DynamoField is powered by Amazon Web Services DynamoDB, a NoSQL database, and DynamoField also provides a front-end interactive web interface. With the flexibility of the NoSQL database, researchers can modify the database schema based on the data provided by various collaborators and contract research organizations. This framework includes functions for non-technical users, including importing and exporting data, data integration and manipulation, and performing statistical analysis. Researchers can utilize cloud computing to establish a secure NoSQL database with minimum maintenance, this also enables collaboration with others worldwide and adapt to different data-collecting strategies as their research progresses. DynamoField is implemented in Python, and it is publicly available at https://github.com/ComputationalAgronomy/DynamoField.

Keywords Field trial, NoSQL database, Web application, DynamoDB, Amazon Web Services (AWS)

The global population is projected to reach 8.5 billion people by 2030 and is expected to increase to 9.7 billion by 2050[1]. As a result of the rapidly growing population, the demand for food production will increase dramatically in the foreseeable future. In addition, the ongoing climate change will also have unpredictable effects on crop production. Researchers from academic and industry sectors share a common goal of addressing these challenges and ensuring we can produce enough food for the fast-growing world population[2–4]. Several potential solutions include improving existing agriculture practices and management systems, breeding new and resilient seed varieties to increase crop yield, and discovering sustainable agricultural biological products to replace chemical products[5].

Researchers and farmers collaborate to develop new agricultural products, including new seed varieties, chemical or biological products such as fertilizers, and farming management practices and techniques[6]. Due to the variation of the natural environment, agricultural products or practices often require additional testing beyond labs and greenhouses. Therefore, rigorous field trials are essential to the validation processes[7,8].

With advances in technology, field trials have collected a large amount of diverse data. These include standard information such as experimental design, treatments, aerial images captured by drone technology, and continuous time series data such as weather information[9]. Collectively, these data provide researchers with information from different points of view and help them make more accurate decisions. Ultimately, these agricultural products tested in field trials will improve crop yields, food security, and sustainability.

As the number of trials increases, recording and organizing a large amount of data becomes more challenging. A Database is one of the most efficient ways to manage and organize field trial data. An efficient usage of the database ensures the reliability of data storage and tracks the relationship between different types of data. At the same time, using a database promotes collaboration among researchers within an organization and makes it easier to share data among farmers and external researchers. Establishing a secure global-scale database capable of facilitating collaboration between many researchers is challenging. This challenge grows when maintaining

[1]Department of Agronomy, National Taiwan University, Taipei, Taiwan. [2]Pivot Bio, Sheboygan, WI, USA. ✉email: stevenwu@ntu.edu.tw

a server with high-security standards. The cost of infrastructure, security and upkeep could impose significant costs for small research teams[10,11].

Common databases paradigms

There are several different paradigms of databases, some commonly used, including flat file, relational, and NoSQL databases[12]. The flat file database is the simplest and easiest to use; all data is stored in a table or spreadsheet format. The advantages of this approach are that it is user-friendly and has a minimum learning curve. However, the disadvantage of this database system is that it does not record the structure of the data. Some widely used spreadsheet programs, such as Microsoft Excel or Google Sheets, provide great user interfaces with this type of database.

The relational database is one of the most widely used databases[13,14]. There are several paradigms and implementations, including both commercial versions, such as Oracle and MS SQL, and open-source alternatives, such as PostgreSQL, MariaDB, and SQLite. A powerful and versatile database requires rigorous database design, and it can be challenging to adapt a new database design rapidly, which requires understanding the underlying relational structure and technical skills to update it. The advantages of the relational database lie in its rigorous and robust frameworks to store large amounts of data and the ability to execute complex queries to retrieve information efficiently.

The NoSQL database, which stands for "Not Only SQL", can go beyond the limitations of traditional relational or SQL-based databases. NoSQL databases are more flexible and can manage unstructured or semi-structured data[15]. In recent years, the NoSQL database has gained popularity and is predominantly used for Big Data analysis[16,17]. There are several sub-types of NoSQL databases; each has its own strength and is tailored to specific use cases. NoSQL databases can be divided into four major categories, and some implementations include features from more than one category[18,19]: (1) Document-oriented databases: Store and retrieve data in flexible and self-describing document formats. Some commonly used formats include JavaScript Object Notation (JSON) and Extensible Markup Language (XML); (2) Key-value database: Store data as key-value pairs, where each value is associated with a unique key; (3) Column-oriented databases: Organize data in columns rather than rows, allowing for efficient retrieval of columns; (4) Graph databases: Represents data as nodes and edges to enable efficient traversal and analysis of complex interconnected data.

Existing database for field trials

Agriculture field trials collect large amounts of data in a wide range of formats across multiple time points. There are several existing solutions for this, most of which are built based on relational databases. There are many software tools, both commercial and open source, available for managing and analyzing agriculture field trial data. Some commonly used commercial software include QuickTrials by RESONANZ Group (https://www.quicktrials.com/), Croptracker by Dragonfly IT (https://www.croptracker.com/), CropTrak (https://www.croptrak.com), Cropwise by Syngenta (https://www.cropwise.com/), and Agriculture Research Management (ARM) by GDM solution (https://gdmdata.com/). These commercial software packages are designed to store and analyze many field trials. They incorporate sophisticated analytic functions for analyzing a diverse range of field trial experiments. Nevertheless, commercial software products are not always the most practical and efficient solution for small research teams or farmers who conduct their own on-farm research trials.

Commercial products typically have a substantial one-time purchasing cost or recurrent annual fee. The licensing agreement might also impose restrictions on the number of concurrent users. Feature-rich software often comes with a steep learning curve and sometimes requires in-depth knowledge of the backend database. Some programs come with specific hardware or infrastructure requirements. Lastly, researchers have limited capabilities to customize the analytic workflow. Some products provide a free version with limited features or a trial version with a limited period.

Alternatively, there are several open-source or non-commercial software that are publicly available, and they are freely available and provide various features for users to record and analyze field trial data. Examples include Agroportal (https://agroportal.lirmm.fr/)[20] and Open Foris (https://openforis.org/)[21], powered by the PostgreSQL database. Some applications also support mobile platforms, such as FieldBook by PhenoApp (https://www.phenoapps.org/)[22]. While storing data in a relational database is a great solution, there are some hurdles for researchers to utilize the database, especially for individual farmers and small-scale research teams.

One limitation of relational databases is the complexity of the underlying data structure, requiring in-depth knowledge to utilize their full capabilities. Users must have a solid understanding of the underlying relationship between each table in the database, which can be challenging. Another limitation due to the nature of field trials is that data provided by farmers and Contract Research Organizations (CROs) often lack consistency. Ongoing updates and maintenance are required to keep up with changes for the latest information. A few common approaches to address these issues include expanding the in-house information technology team, or out-source to external developers. While these are feasible solutions for large-size companies, they may not be optimal or practical for small-scale projects. Therefore, it is common for small-scale projects to record data in spreadsheet format, using software such as Microsoft Excel, Google Sheets, or Apple Numbers. These user-friendly software provide essential features for collecting and analyzing small amounts of data, especially when all data can be contained in a single spreadsheet.

Amazon Web Services and DynamoDB

Amazon Web Services (AWS) is one of the leading cloud computing service providers. Combined with other leading service providers, Microsoft Azure and Google Cloud, they hold more than 60% of the market share[23,24]. DynamoDB is a NoSQL database developed by AWS. It is a key-value database that is designed to provide versatile and flexible database services. AWS hosts the DynamoDB on the cloud and provides a highly scalable

service that enables users to work collaboratively online. AWS also released the standalone version of DynamoDB for users to host their own database locally. DynamoDB allows a very flexible framework, and the underlying data schema can be updated on the fly. This is ideal for projects that frequently change and evolve, or need to adapt to new requirements quickly[25].

However, DynamoDB presents its own challenges for non-technical end users. The DynamoDB services itself does not provide a web-based interface for managing large amounts of data. There are several user interfaces available, such as AWS NoSQL Workbench or Dynobase. However, they require a solid understanding of the NoSQL database to constitute the database, or they are subscription-based. The primary method of interacting with DynamoDB is through AWS's application programming interfaces (API), which are available in multiple programming languages, including Java, Python, Go, etc. The data requires to be converted to a specific JSON format in order to be imported to DynamoDB[25]. JSON format organizes data using a hierarchy structure with nested bracket pairs, the data is defined by name-value pairs[26]. This format is excellent for machines to transfer information, but it is challenging for humans to read and write manually. These hurdles impose significant challenges for researchers, agronomists, farmers, and CROs to utilize the DynamoDB database in a collaborative manner. For more information on additional features of DynamoDB, please refer to the supplementary material.

In this paper, we present DynamoField, a flexible NoSQL database framework designed specifically for small- and medium-scale agriculture field trials with a user-friendly web interface that enables collaborators around the world to access it on the cloud. The backend NoSQL database is powered by the DynamoDB developed by AWS, and utilizing the AWS global network to manage all servers and security for end users. Researchers across the globe can access and query the DynamoDB database. Although utilizing the NoSQL database and cloud computing in agriculture is not new, there are very few publicly available database schemas using NoSQL[27,28]. This schema allows end users to utilize the capabilities of the database, without designing and committing to the underlying relationship between different types of data. The highly flexible structure can adapt to various changes in protocols or CROs during multi-year field trials. The web-based user-friendly interface enables all researchers to utilize this flexible database without any technical knowledge of the NoSQL database. Users can search and query the databases without manually building complex queries in JSON format. In addition to storing data, this web interface is also capable of performing basic statistical analyses for field trial data. For other sophisticated or customized statistical analyses, researchers can export the data and analyze it with their own analytic pipeline.

Materials and methods

Field trial data

A very wide range of data can be collected depending on the purpose of an agriculture field trial. A simple trial may have minimal information or data collected, and more complex trials may have large amounts of information collected. The information collected can be divided into multiple different categories based on the nature of the data. Some are directly related to the actual trial, and often, these are predefined before trials are conducted, for example, the variety or cultivar of the seed and treatments that will be tested in each trial. Other information related to the entire trial, includes the physical location of the trial, GPS locations, owner or manager's contact information, and so on. The majority of these data can be stored in categorical or semi-structured format depending on the nature of the data, whereas GPS locations have a few different standard formats, including decimal and degree formats. These are likely to remain constant throughout the trial. Throughout the field trial season, additional information is collected at a trial or plot level, such as the farm management practices, irrigation systems and amounts, time and the amount of fertilizer or pesticides applied, soil analyses. This information is likely to come in free text format, and each farmer and CRO will have different recording systems. The intermediate reports regarding the plant population, yield, or disease status can be in numerical or categorical formats. Some data are collected at multiple different time points, which can be on regular or non-regular time intervals depending on the purpose of the trial. For example, weather, rainfall, and temperature data would benefit from time series analysis. These data are often collected systematically with a predefined format from the machine. At the end of the field trial season, if the trial is harvested, the final yield often concludes the trial data collection.

Field trial data and results are provided by multiple different CROs, university researchers, farmers, or from internal research. Data would be collected at multiple different time points, depending on the trial locations and the purpose of the trial. It is not always feasible to request all parties to provide data in identical format across all trials. Therefore, one major challenge for researchers is to transform raw data contributed from multiple parties to the same format in order to fit the structure of the relational database.

DynamoField

The DynamoField we proposed is a flexible framework with a graphical user interface that enables researchers to import data with minimum preprocessing. DynamoField utilizes the flexibilities of the NoSQL database paradigm, to transfer and integrate the data. This will reduce the amount of data preprocessing before importing it into the database and can incorporate various types of data from different parties. The database can handle a wide range of storage types, ranging from simple data type, such as numbers and characters, to more complicated data type, such as lists and maps. The database supports converting from a simpler storage type to a more complex one.

DynamoField uses DynamoDB as its backend NoSQL database, and it supports both cloud and local versions. The cloud version is a fully managed service on the cloud, where users do not need to maintain a local server. The cloud version also allows collaborators to access the database across the globe. The local version of the DynamoDB database can be deployed to an in-house server without depending on or paying for cloud computing. Users have full control with this version and developers are able to test applications locally. The

local version runs with Java Runtime Environment (JRE) or a Docker container. There are several technical differences between the local and cloud versions of the database, and the details are described in the Amazon DynamoDB Documentation[25].

The database framework we proposed is based on a few key requirements. This database allows users to create and modify the database schema on the fly, therefore users do not have to commit to a specific data structure. The database utilizes a composite key schema consisting of two keys: a partition key and a sort key;

- The partition key is the unique identifier for each field trial, this database will refer to this as *field_trial_ID*.
- The sort key is designed to store various information and different types of data associated with each field trial. The NoSQL database provides great flexibility and users do not have to impose a set of fixed rules on specifying the sort keys. In addition, sort keys do not have to be the same between field trials. This database schema will refer to this as *record_type*.

There are two major advantages of this approach. This establishes a generic and flexible framework for integrating a wide range of data. The composite key schema can incorporate new data types by adding new sort keys to the database. There are no limits to the number of different sort keys allowed to the database. In addition, this enables cost-effective usage of the database, as cloud computing can reduce IT personal and infrastructure costs. An example of the usage of a composite key in the database is illustrated in Fig. 1A, and an example of the AWS DynamoDB JSON format is shown in Fig. 1B. The schema presented here is designed for agriculture field trials. Nevertheless, it can be adapted for similar types of experiments, such as trials performed in growth chambers and greenhouse experiments. These experiments can have the same experimental designs, materials, treatments, and collect comparable data at the end of experiment.

The partition key, *field_trial_ID*, is required for every data point. The content of this identifier does not need a fixed naming schema as long as each trial can be uniquely identified. Researchers can determine their preferred method of specifying the *field_trial_ID* with their own naming convention. The scope of an agricultural field trial is generally determined by specific objectives and other parameters. Here, we defined a field trial as a self-contained experiment, typically conducted at one single location with a fixed number of treatments. Most of the field trials are associated with an experiment design. This could be the Completely Random Design (CRD), widely used Random Complete Block Design (RCBD), or any other used design. This unique *field_trial_ID* is used for grouping all information associated with a specific field trial together, including seed information, treatment information, locations and farm management, weather, etc.

Within each field trial associated with a unique *field_trial_ID*, researchers collected a large amount of data that comprised multiple different data types, including the layout of experimental design, treatments of interest, location and contact information of the field trial, weather, final outcome of the trial, etc. Information collected for field trials is likely to change across different locations, CROs, and between years. There are several reasons that can contribute to this, including the scope and purpose of the trials, which can change over time. The amount of information collected is likely to increase as the project moves from a pilot study or proof of concept to large-scale validation studies. Lastly, trials may be conducted by different CROs in different years due to unforeseen and uncontrollable circumstances.

The DynamoField database framework organizes all data into multiple *record_types*, and each stores a different type of information. Furthermore, there is no requirement for each field trial to have the same number of data types, and not all data types are mandatory for every field trial.

(A)

Composite Key						
Partition key	**Sort Key**					
field_trial_id	record_type	Crop	Treatment	Owner	Farming type	Irrigation system
Trial_1	Treatment_0	Corn	Control			
Trial_1	Treatment_1	Corn	Fertilizer 50ppm			
Trial_2	Treatment_0	Corn	Control			
Trial_2	Treatment_1	Corn	Fertilizer 50ppm			
Trial_3	Treatment_0	Sorghum	Control			
Trial_3	Treatment_1	Sorghum	Fertilizer 50ppm			
Trial_3	Treatment_2	Sorghum	Fertilizer 100ppm			
Trial_1	Location_0			John Doe	Organic	
Trial_2	Location_0			Jane Doe	Conventional	
Trial_3	Location_0			NA	Conventional	
Trial_2	Irrigation_0					Drip
Trial_3	Irrigation_0					Drip

(B)


```
{
  "Item": {
    "field_trial_id": { "S": "Trial_1" },
    "data_type": { "S": "treatment_0" },
    "Crop": { "S": "Corn" },
    "Treatment": { "S": "Control" }
  },
  "Item": {
    "field_trial_id": { "S": "Trial_1" },
    "data_type": { "S": "treatment_1" },
    "Crop": { "S": "Corn" },
    "Treatment": { "S": "Fertilizer 50ppm" }
  },
  "Item": {
    "field_trial_id": { "S": "Trial_2" },
    "data_type": { "S": "treatment_0" },
    "Crop": { "S": "Corn" },
    "Treatment": { "S": "Control" }
  },
  "Item": {
    "field_trial_id": { "S": "Trial_2" },
    "data_type": { "S": "treatment_1" },
    "Crop": { "S": "Corn" },
    "Treatment": { "S": "Fertilizer 50ppm" }
  }
}
```


Fig. 1. (**A**) Example of the usage of composite keys, partition keys: *field_trial_ID* and sort keys: *record_type*. The sort keys highlighted in yellow show that each partition key can have a variable number of samples for a sort key. The sort keys highlighted in blue show that missing values (NA) are allowed. The sort keys highlighted in green show not all sort keys are required for each partition key. (**B**) Example of the AWS DynamoDB JSON format.

(A)

Composite Key		Attributes					
Partition key	**Sort Key**						
field_trial_id	data_all_weather	GPS location	Sites	Rainfall	Temperature daily high	Temperature daily average	Temperature daily low

(B)

Composite Key		Attributes		
Partition key	**Sort Key**			
field_trial_id	weather_rainfall	GPS location	Sites	Rainfall

Composite Key		Attributes				
Partition key	**Sort Key**					
field_trial_id	weather_temperature	GPS location	Sites	Temperature daily high	Temperature daily average	Temperature daily low

Fig. 2. Example of flexible *record_types* creation for the database. The blue color represents the partition key, and the green color represents the sort key, which together form the composite key in the database. (**A**) Use a single sort key to collect all data relate to weather. (**B**) Users can break it into multiple sort keys, one for the rainfall and the other for temperature. These columns can be combined with the merging feature provided by DynamoField.

DynamoField

Composite Key				
Partition key	**Sort Key**			
field_trial_id	record_type	CRO	Year	Irrigation records
Trial_1	Irrigation_0	CRO_A	1	Yes
Trial_2	Irrigation_0	CRO_A	1	No
Trial_3	Irrigation_0	CRO_B	2	150 units
Trial_4	Irrigation_0	CRO_B	2	90 units
Trial_5	Irrigation_0	CRO_C	3	Drip
Trial_6	Irrigation_0	CRO_C	3	Sprinkler
Trial_7	Irrigation_0	CRO_C	3	Drip

Relational Database

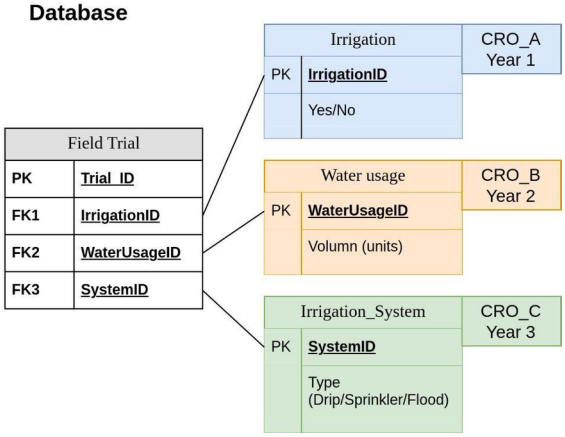

Fig. 3. Illustration of DynamoField's flexibility in accommodating data from multiple CROs across different years. DynamoField utilizes a composite key structure allowing seamless incorporation of new data each year without modifying the database structure. In contrast, relational databases require updating the schema and creating new tables for each CRO to accommodate varying data formats and types, such as boolean, integer, and categorical data.

The scope of each *record_type* is defined by the research team. One can opt for a generic data type that encompasses a wide range of information. For example, one can have the "data_all_weather" type that includes everything related to weather, such as temperature, rainfall, humidity, wind, date, and time stamp for each record. This approach is illustrated in Fig. 2A. Alternatively, one can split this data into multiple different *record_types*. This can be beneficial when different data are generated from different instruments with different time periods. Researchers do not have to aggregate data prior to importing them into the database. For example, the "weather_rainfall" data type only stores date information and the amount of rainfall per day, "weather_temperature" only stores daily records for maximum, minimum, and average temperature, and "weather_wind" only stores time information and wind speed and direction. This alternative approach is illustrated in Fig. 2B. Regarding the technical implementation of the DynamoField, the *record_type* acts as the sort key in the database. Along with the field trial ID, which is the partition key, these two keys are combined together to form the composite key. When multiple data points with the same field_trial_ID and same *record_type*, DynamoField will automatically append a non-duplicate suffix to the sort key field to ensure the uniqueness of the composite key.

Third-party collaborators, such as farmers or contract research organizations (CROs), conduct field trials and present the data in various formats. In addition, these third-party collaborators might change over the years, leading to different data formats across multiple years. Figure 3 illustrates the flexibility of DynamoField,

where new data from new collaborators each year can be incorporated into the database without modifying the database structure. On the other hand, relational databases require updating the schema and underlying structures in order to incorporate new data for each new collaborator, which can be time-consuming and require technical expertise.

Security

Data security is critical for maintaining the database[29]. The security of DynamoField depends on the backend NoSQL database. For the local version, the access to the database is controlled by the local administration. DynamoField does not have separate accounts with different read and write permissions. Once the access is granted, users have full access to the database. For the cloud version of the database, DynamoField uses the AWS access token to control the access and usage. The in-house AWS account owners and administrators have full control over the access, reading, and writing permissions of each user. In addition, admins have the authority to revoke access at any time.

Results

Implementation

We developed DynamoField, a flexible database framework for collecting and analyzing field trial data. DynamoField provides a user-friendly front-end web interface, which removes the technical burden of accessing the backend NoSQL database. The interface for DynamoField is illustrated in Fig. 4, it is divided into three core sections and represented by three tabs. The server end "Database status" tab manages the connection to the backend database for both local and cloud servers, and queries and creates new tables. (Fig. 4A). The "Import data" tab lets users specify the *record_type*, and imports data from Excel and CSV formats (Fig. 4B). The "Query database" tab allows users to perform flexible queries using composite keys and other fields, retrieve data from the database, perform multiple data manipulations, and basic statistical analysis (Fig. 4C). In addition, users can export the full dataset or query results to CSV format and subsequently perform customized statistical analyses tailored to their experiments.

DynamoField is implemented in Python, which is available on all major operating systems.

The front-end web interface is built with the Dash framework. This framework handled all users' inputs by reactive programming and triggered callback functions to perform various operations. These operations included connecting to the database, modifying the database tables, importing and exporting data, querying and searching data from the database, etc. The communication between the interface and the NoSQL database is handled by the Boto3 library. DynamoField includes built-in tests to ensure its accuracy and reliability, which maintains the quality of the software. The software architecture of this package is summarized in Fig. 5.

Import, query, and export

In order to import data into the database, DynamoField allows users to upload data in either Excel or CSV formats. Subsequently, it performs the data transformation and converts data into JSON format for end users. The default storage type for all data is characters, and the database supports users in modifying the storage type at any time. In addition, DynamoField performs several essential checks, verifying the presence of *field_trial_ID* and *record_type*, which serve as partition and sort keys in the database, respectively. It also checks for the uniqueness of the composite key, and a distinct numerical suffix is appended to each *record_type* to ensure uniqueness for each key. DynamoField uses the Boto3 package to communicate and import the data into the database. DynamoField enables users to dynamically generate new keys on-the-fly during data import, this offers researchers the flexibility to introduce new *record_type* and store a wide range of data.

DynamoField provides multiple methods for users to query and retrieve data from the database. The most efficient operation is directly querying a known partition key, sort key, or composite key. Querying the partitioning key allows users to retrieve all information related to one or multiple field trials. On the other hand, querying the sort key alone can retrieve all data from the same data type. Querying the composite key allows users to retrieve information for a specific data type for a trial. In addition, DynamoField also provides the query feature to search any fields in the database. The query feature supports wild cards and partial text, allowing users to retrieve information using any criteria. DynamoField utilizes both the query and scan features and retrieves all information in JSON format, subsequently converting it into a human-readable table format. DynamoField provides the exporting function, and users can export the entire dataset, or query results to Excel or CSV format for downstream analysis. Both importing and exporting processes are illustrated in Fig. 6.

Data manipulation and integration

DynamoField provides multiple data manipulation and integration features. Several useful operations include merging multiple columns within the same *record_type* and merging data from different *record_types* together. These features reduce the burden of data preprocessing while data are provided by various CROs, university researchers, farmers, or internal researchers. All data can be imported into the DynamoDB database as it is. Data manipulation and merging data from multiple *record_types* can be performed in the database, and this feature is illustrated in Fig. 7. The merging process creates a new *record_type*, which concatenates two existing *record_types* together, this is used to prevent the loss of information. During the merging process, if two columns with exactly the same name and have identical records, these records will be combined together. This mimics the "join" operation in the relational database.

Large-scale field trials involve multiple parties, who could have very different practices and data collection workflow. Therefore, the naming conventions and data formats are likely inconsistent between these data providers. DynamoDB database allows each party to import the data into the database with their own naming conventions. Subsequently, researchers can merging multiple columns within the same *record_type* in the

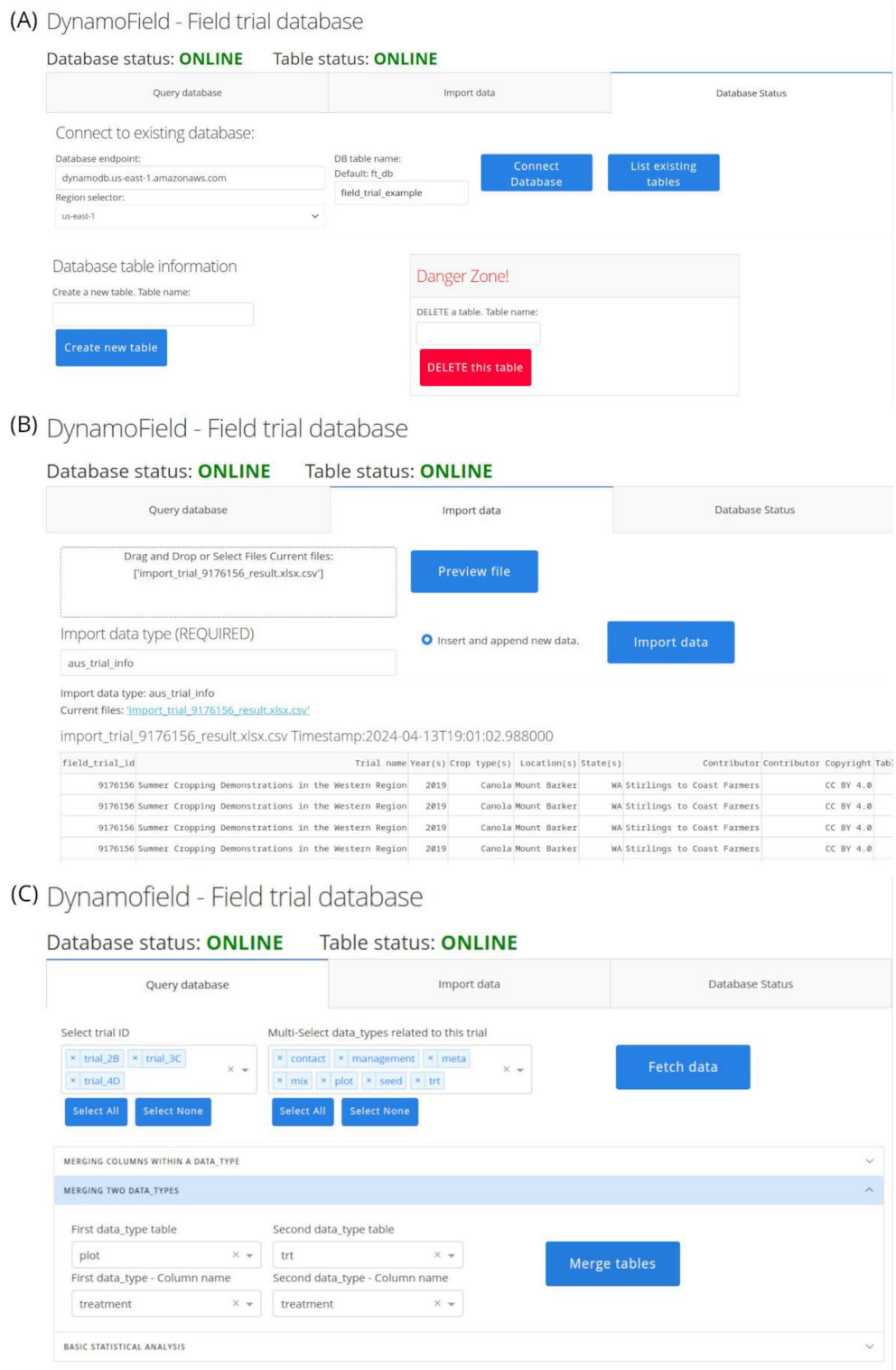

Fig. 4. Demonstrate the user interface for DynamoField. (**A**) The "Database status" tab, allows users to connect to the database. (**B**) The "Import" tab, allows users to import data and create new composite keys. (**C**) The "Query" tab, queries and retrieves data from the database, performes merge columns and other data manipulation, statistical analysis, and exporting data.

database, this feature is illustrated in Fig. 8. The merging process is takes into account the composite keys, including both trial information and *record_type*, which can prevent conflicts to occur. In addition, the merging process will fail if two non-empty cells are merged together, users have to determine which datapoint do they want to use in the database.

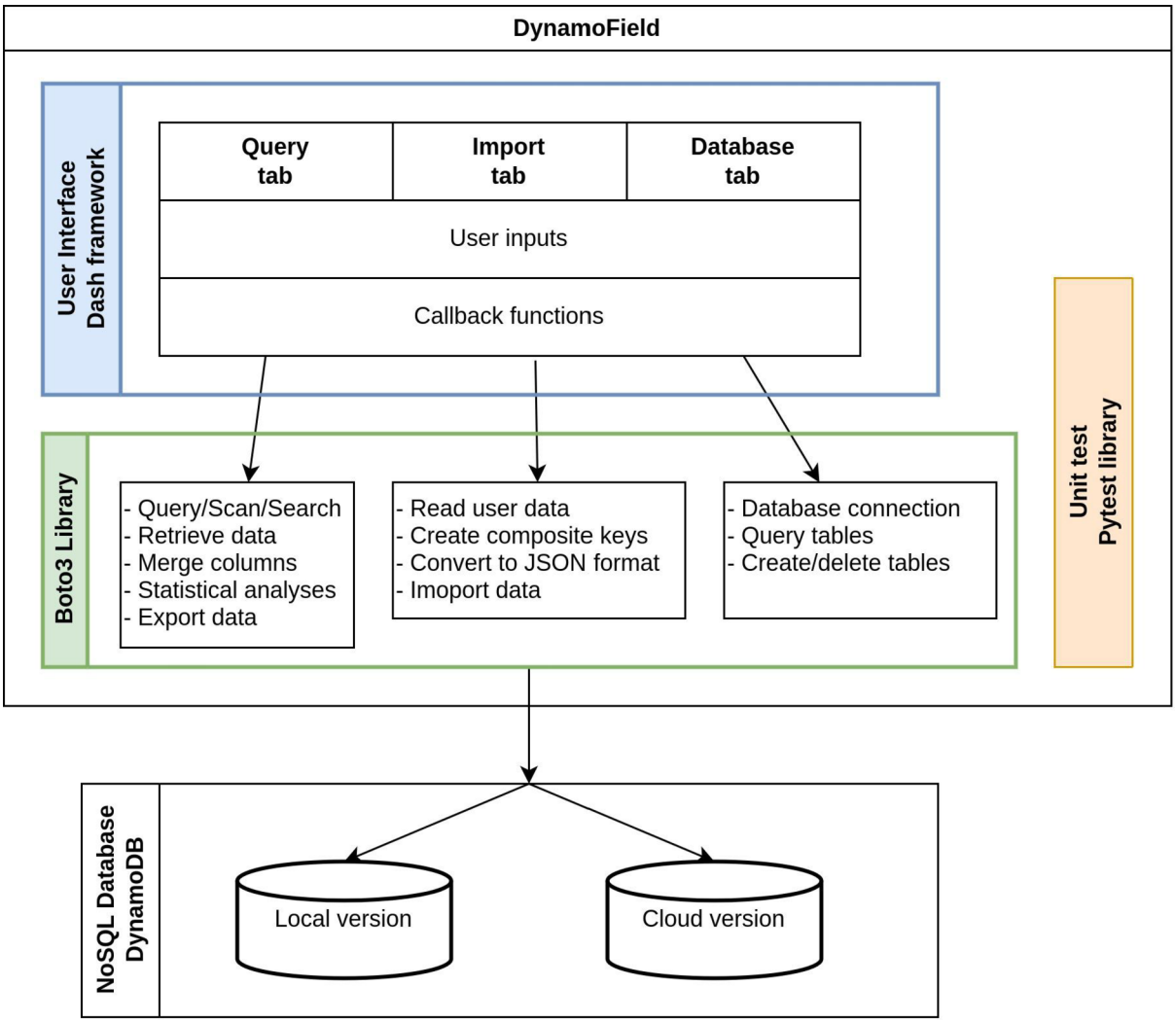

Fig. 5. Architecture for the Python implementation for DynamoField and communication to the NoSQL database. The blue box summarizes the user interface and actions are captured and passed to the next stage by the callback functions. The green box summarizes the underlying database operation performed by the Boto3 library. This library also handles the communication between DynamoField and the database. The orange box represents the testing framework shipped with this package.

Application to the public dataset

DynamoField demonstrated its capability by integrating data from multiple field trials from the Online Farm Trials (OFT) hosted by Grains Research and Development Corporation (GRDC), Australia[30]. Seventeen farm trial projects were selected between 2013 and 2023 that satisfied the searching criteria "all cereal crops" and under Creative Commons license CC BY 4.0. The number of trials per project range between 1 and 31, and the number of features collected ranges from 17 to 62. The list of these projects can be found in Table S1. There are only 16 columns shared between all projects, such as"'Trial ID", "Trial name", and 'Contributor', etc. DynamoField allows each project to be imported independently. Subsequently, data manipulation, merging, and rearrangement of data from multiple projects can be performed within the database. In addition, queries across different projects and retrieving data for further analysis. These projects are included as example datasets with the DynamoField package on GitHub.

Discussion

With the improvement in technologies, researchers are able to collect more data for field trials. We are in the era of big data, and the NoSQL database is capable of handling huge amounts of data. NoSQL database schema can be adopted by large multinational projects with thousands of field trials across the world.

The flexibility of DynamoField provides researchers with the ability to modify the database as the project evolves and expands. The research teams do not have to predict future growth and potential data collection methods. The NoSQL database schema can be updated on the fly, and users are not required to commit to a predetermined database structure. This flexibility enables the incorporation of unexpected events, such as

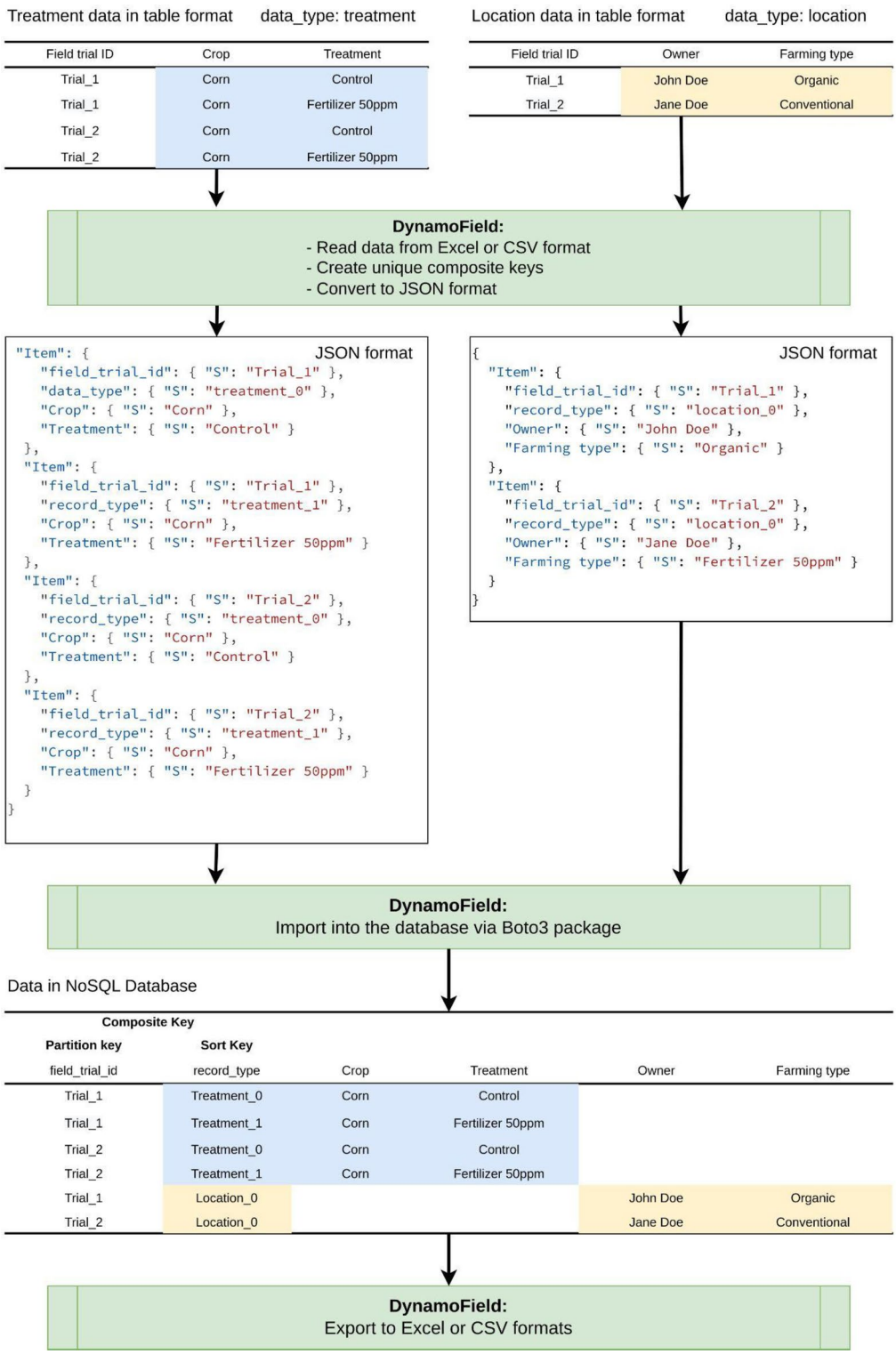

Fig. 6. Demonstrates the import and export process of DynamoField. The original data is in the basic table format, such as MS Excel or CSV format. The blue dataset is imported with *record_type* treatment, and the yellow dataset is imported with *record_type* location. The DynamoField converts the data from table format to JSON format and then imports it into the backend database. Subsequently, it can be exported for further analysis.

Treatment data in DynamoDB

Composite Key				
Partition key	**Sort Key**			
field_trial_id	record_type	Crop	Treatment	Location
Trial_1	Treatment_0	Corn	Control	Location_A
Trial_1	Treatment_1	Corn	Fertilizer 50ppm	Location_A
Trial_1	Treatment_2	Corn	Fertilizer 100ppm	Location_A
Trial_1	Treatment_3	Corn	Fertilizer 200ppm	Location_A
Trial_2	Treatment_0	Corn	Control	Location_B
Trial_2	Treatment_1	Corn	Fertilizer 50ppm	Location_B
Trial_2	Treatment_2	Corn	Fertilizer 100ppm	Location_B
Trial_2	Treatment_3	Corn	Fertilizer 200ppm	Location_B

Location data in DynamoDB

Composite Key				
Partition key	**Sort Key**			
field_trial_id	record_type	Owner	Phone number	Farming type
Trial_1	Location_A	John Doe	000-123456789	Organic
Trial_2	Location_B	Jane Doe	000-987654321	Conventional

Merged Treatment and Location data in DynamoDB

Composite Key						
Partition key	**Sort Key**					
field_trial_id	record_type	Crop	Treatment	Owner	Phone number	Farming type
Trial_1	Treatment_Location_0	Corn	Control	John Doe	000-123456789	Organic
Trial_1	Treatment_Location_1	Corn	Fertilizer 50ppm	John Doe	000-123456789	Organic
Trial_1	Treatment_Location_2	Corn	Fertilizer 100ppm	John Doe	000-123456789	Organic
Trial_1	Treatment_Location_3	Corn	Fertilizer 200ppm	John Doe	000-123456789	Organic
Trial_2	Treatment_Location_0	Corn	Control	Jane Doe	000-987654321	Conventional
Trial_2	Treatment_Location_1	Corn	Fertilizer 50ppm	Jane Doe	000-987654321	Conventional
Trial_2	Treatment_Location_2	Corn	Fertilizer 100ppm	Jane Doe	000-987654321	Conventional
Trial_2	Treatment_Location_3	Corn	Fertilizer 200ppm	Jane Doe	000-987654321	Conventional

Fig. 7. Illustration of merging data from two *record_types*, Treatment and Location, to create a new *record_type* called Treatment_Location, highlighted in blue. During the merging process, the location data, highlighted in green and yellow, are merged with the respective rows in the treatment data.

changes in CROs and rare natural disasters. The graphical user interface removes the burden of learning the technical details required to operate the backend database. The web interface also implemented several end-user-oriented features, including data manipulation, exploratory analysis, plotting functions, and exporting functions for sophisticated analysis. In addition, DynamoField is open-source software implemented in Python, a cross-platform language available on any machine. This program can be run on most personal computers purchased within the last five years. We recommend that users run DynamoField on a computer with at least 16GB of RAM, especially if they are working with a large amount of data.

For the backend NoSQL database, there are cloud and local versions of DynamoDB available for end users. The cloud version removes the burden of managing local servers and the resources associated with the IT team. However, if the research team stores large amounts of images or videos, extra costs will be accumulated depending on the data size. The local version is implemented in Java, which is also cross-platform and can be hosted on any local server. There are some drawbacks to hosting a local version of the NoSQL database. The local version has limited scalability and does not support parallel scan operations, making it less suitable for handling large workloads. Users are required to set up their own server infrastructure for the database. The exact cost and specifications of the server depend on the scope of the company and other factors such as the number of local users, remote collaborators, and backup strategies.

Conclusion

We present DynamoField, an ultra-flexible NoSQL database framework designed to collect, organize, and analyze agricultural field trial data. The web user interface removes the burden of learning database syntax format and technical details for end users. DynamoField can modify the data collection schema and incorporate a wide range of data types that are collected by multiple CROs and collaborators. Most of the existing software has yet to utilize the full potential of the NoSQL database. In the era of big data, the NoSQL database provides the flexibility and scalability to organize huge amounts of data.

Data imported by multiple CROs

Composite Key					
Partition key	**Sort Key**				
field_trial_id	record_type	CRO	Irrigation	Water usage	Irr usage
Trial_1	Management_0	CRO_A	Drip		
Trial_1	Management_1	CRO_B	Drip		
Trial_1	Management_2	CRO_C	Drip		
Trial_2	Management_0	CRO_D		None	
Trial_2	Management_1	CRO_E		None	
Trial_2	Management_2	CRO_A		Drip	
Trial_3	Management_0	CRO_F			None
Trial_3	Management_1	CRO_A			Sprinkler
Trial_3	Management_2	CRO_E			Sprinkler

Merged multiple columns within the same data type

Composite Key						
Partition key	**Sort Key**					
field_trial_id	record_type	CRO	Irrigation Water System	Irrigation	Water usage	Irr usage
Trial_1	Management_0	CRO_A	Drip	Drip		
Trial_1	Management_1	CRO_B	Drip	Drip		
Trial_1	Management_2	CRO_C	Drip	Drip		
Trial_2	Management_0	CRO_D	None		None	
Trial_2	Management_1	CRO_E	None		None	
Trial_2	Management_2	CRO_A	Drip		Drip	
Trial_3	Management_0	CRO_F	None			None
Trial_3	Management_1	CRO_A	Sprinkler			Sprinkler
Trial_3	Management_2	CRO_E	Sprinkler			Sprinkler

Fig. 8. Illustrate merging multiple columns within the same *record_type*, management. These three columns, Irrigation, Water usage, and "Irr usage" were imported by three different CROs. The new column "Irrigation Water System" is highlighted in blue and combines information from different CROs in the NoSQL database.

Data availability

The datasets generated during and/or analysed during the current study are available in the GitHub - DynamoField repository, https://github.com/ComputationalAgronomy/DynamoField.

Received: 14 April 2024; Accepted: 14 November 2024
Published online: 30 November 2024

References

1. United Nations Department of Economic and Social Affairs Population Division. World PopulationProspects 2022: Summary of Results (2022).
2. Capalbo, S. M., Antle, J. M. & Seavert, C. Next generation data systems and knowledge products to support agricultural producers and science-based policy decision making. *Agric. Syst.* **155**, 191–199 (2017).
3. Kamilaris, A., Kartakoullis, A. & Prenafeta-Boldú, F. X. A review on the practice of big data analysis in agriculture. *Comput. Electron. Agric.* **143**, 23–37 (2017).
4. Basso, B. & Antle, J. Digital agriculture to design sustainable agricultural systems. *Nat. Sustain.* **3**, 254–256 (2020).
5. Silva, J. V. & Giller, K. E. Grand challenges for the 21st century: what crop models can and can't (yet) do. *J. Agric. Sci.* **158**, 794–805 (2020).
6. Laurent, A., Kyveryga, P., Makowski, D. & Miguez, F. A Framework for visualization and analysis of agronomic field trials from On-Farm Research Networks. *Agron. J.* **111**, 2712–2723 (2019).
7. Forero, L. E., Grenzer, J., Heinze, J., Schittko, C. & Kulmatiski, A. Greenhouse- and field-measured plant-soil feedbacks are not correlated. *Front. Environ. Sci.* **7**, 184 (2019).
8. Zeller, S. L., Kalinina, O., Brunner, S., Keller, B. & Schmid, B. Transgene × environment interactions in genetically modified wheat. *PLoS One* **5**, e11405 (2010).
9. Cheng, Y. T., Zhang, L. & He, S. Y. Plant-microbe interactions facing environmental challenge. *Cell Host Microbe* **26**, 183–192 (2019).
10. Wolfert, S., Ge, L., Verdouw, C. & Bogaardt, M. Big data in smart farming—a review. https://doi.org/10.1016/j.agsy.2017.01.023 (2017).
11. Holland, R., Khanal, A. R. & Dhungana, P. Agritourism as an alternative On-Farm enterprise for small U.S. farms: examining factors influencing the agritourism decisions of small farms. *Sustainability* **14**, 4055 (2022).

12. Khan, W. et al. SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review. *Big Data Cogn. Comput.* **7**, 97 (2023).
13. DB-Engines Ranking. https://db-engines.com/en/ranking (2023).
14. TOPDB Top Database index. https://pypl.github.io/DB.html (2023).
15. Vera-Olivera, H. et al. Data modeling and NoSQL databases—a systematic mapping review. *ACM Comput. Surv.* **54** (2021).
16. Faridoon, A. & Imran, M. Big Data Storage Tools using NoSQL databases and their applications in various domains: a systematic review. *Comput. Inf.* **40**, 489–521 (2021).
17. Martinez-Mosquera, D., Navarrete, R. & Lujan-Mora, S. Modeling and management big data in databases—a systematic literature review. *Sustainability* **12**, 634 (2020).
18. Candel, C. J. F., Ruiz, D. S. & García-Molina, J. J. A unified metamodel for NoSQL and relational databases. *Inf. Syst.* **104**, 101898 (2022).
19. Chen, L., Davoudian, A. & Liu, M. A workload-driven method for designing aggregate-oriented NoSQL databases. *Data Knowl. Eng.* **142**, 102089 (2022).
20. Jonquet, C. et al. AgroPortal: a vocabulary and ontology repository for agronomy. *Comput. Electron. Agric.* **144**, 126–143 (2018).
21. *Open Foris Initiative of the FAO* (2023).
22. Rife, T. W. & Poland, J. A. Field book: an open-source application for field data collection on android. *Crop Sci.* **54**, 1624–1627 (2014).
23. Synergy Research Group. Q1 Cloud Spending Grows by Over $10 Billion from 2022; the Big Three Account for 65% of the Total. https://www.srgresearch.com/articles/q1-cloud-spending-grows-by-over–10-billion-from–2022-the-big-three-account-for–65-of-the-total (2023).
24. Gartner, I. Gartner ReportMagic Quadrant for Cloud AI Developer Services. https://pages.awscloud.com/Gartner-Magic-Quadrant-for-Cloud-AI-Developer-Services.html?Languages=English (2023).
25. Amazon Web Services. Amazon DynamoDB Developer Guide. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html (2023).
26. Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format (2017).
27. Karmas, A., Tzotsos, A. & Karantzalos, K. Geospatial big data for environmental and agricultural applications. In *Big Data Concepts, Theories, and Applications* 353–390. https://doi.org/10.1007/978-3-319-27763-9_10 (Springer International Publishing, 2016).
28. Nyoman Kutha Krisnawijaya, N., Tekinerdogan, B., Catal, C. & van der Tol, R. Data analytics platforms for agricultural systems: a systematic literature review. *Comput. Electron. Agric.* **195**, 106813 (2022).
29. Amazon Web Services. AWS Well-Architected Framework. https://docs.aws.amazon.com/wellarchitected/latest/framework/the-pillars-of-the-framework.html (2023).
30. Federation University Australia and GRDC. Centre for eResearch and Digital Innovation. *Online Farm. Trials*. https://www.farmtrials.com.au/ (2024).

Acknowledgements

We thank two anonymous reviewers and the editors for their feedback and suggestions, which have greatly improved the quality and clarity of this work. This work was supported by the Ministry of Science and Technology, Taiwan, under the grant number MOST 111-2313-B-002-007-MY3.

Author contributions

S.W. and T.M. conceptualized the idea, validated the software, and wrote the manuscript. S.W. developed the software and analysis the data.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-81609-2.

Correspondence and requests for materials should be addressed to S.H.W.