UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Initial Database Architecture

Databases II

César Andrés Torres Bernal
20191020147
Juan David Duarte Ruiz
20191020159

Engineering Faculty

Initial Database Architecture

To propose a robust and scalable initial database architecture for AgroClima, considering the requirements for big data and distributed databases, it is essential to design a system capable of handling large volumes of geospatial and time-series data from various sources. The platform must support continuous data ingestion, efficient processing to generate real-time alerts, the generation of historical reports, and the execution of complex models for climate and agricultural predictions.

1. Data Sources:

- External APIs: Integration with global providers such as OpenWeather, OpenMeteo and NOAA.
- Historical Climate Data: Retrospective datasets from various sources for trend analysis and model training.

2. Data Ingestion Layer:

This layer is responsible for collecting data from diverse sources, which can vary in format, velocity, and structure. Given the big data nature, a scalable and fault-tolerant ingestion system is required.

- Apache Kafka or Amazon Kinesis: For ingesting real-time and streaming data from weather stations. They allow decoupling data sources from processing systems.
- ETL Tools: For processing data in batches from historical files, climate models, or satellite imagery. Examples: Apache NiFi, Talend, or cloud services like AWS Glue or Google Cloud Dataflow.

3. Distributed Data Storage Layer:

This layer is the responsible to storage all the information, information both raw and already processed for use, to handle the diversity and volume of data, a combination of distributed databases is proposed, each optimized for different data types

- Data Lake: To store raw data from various sources before being processed and refined.
    - Technologies: Apache Hadoop HDFS, Amazon S3, Google Cloud Storage, Azure Data Lake Storage. They provide scalable and cost-effective storage.
- Distributed Storage: Processed data will be stored in a distributed database such as Apache Cassandra, Google Bigtable, or Amazon DynamoDB.

4. Distributed Data Processing Layer:

This layer is responsible for transforming, analyzing, and deriving valuable information from the stored data, supporting both real-time and batch processing.

Technologies: Apache Spark / Apache Flink:

- Handles both batch and stream processing.
- Cleans, transforms, and prepares data for storage and analysis.

5. Serving and Application Layer:

This layer interacts directly with end-users (logistics and agricultural companies), providing access to alerts, reports, and predictions through a user interface or APIs.

- Serving Database: Databases optimized for fast and concurrent reads by the application. These can be materialized views or aggregations of processed data.
    - Suggested Technologies: Relational databases (PostgreSQL, MySQL) for structured data like reports and user profiles, and key-value or document NoSQL databases for fast access data (Redis, MongoDB).
- API Services: To expose platform functionalities to external applications or the user interface.

6. Access and API Layer REST and GraphQL APIs:

Serve data securely to mobile and web clients. Provide modular endpoints for different user roles (farmers, analysts, admins). The Access and API Layer in AgroClima provides secure and scalable interaction between the platform's backend and its user-facing applications. Using REST and GraphQL APIs, the system exposes modular and role-specific endpoints that allow farmers, analysts, and administrators to access data and services tailored to their needs.

ER Diagram (Initial Version)

Below is a description of each entity, with its respective attributes, for the initial version of the ER (Entity-Relationship) diagram for the AgroClima project.

User

Description: Represents the individuals registered in the AgroClima platform. Users may have different roles such as farmers, analysts, or administrators and are the recipients of climate alerts and predictions.

Attributes:

- user_id (INT, PK): Unique identifier of the user.
- user_name (VARCHAR(50)): Full name of the user.
- email (VARCHAR(100), UNIQUE): Email address used for login and notifications.
- role (VARCHAR(50)): User role (e.g., Farmer, Analyst, Admin).
- password_hash (VARCHAR(50)): Encrypted password for authentication.

Alert

Description: Stores records of alerts sent to users when a relevant climate event is predicted. Alerts are based on predictions generated by the system.

Attributes:

- alert_id (INT, PK): Unique identifier of the alert.
- user_id (INT, FK): User who receives the alert.
- prediction_id (INT, FK): Prediction that triggered the alert.
- timestamp_sent (DATETIME): Date and time the alert was sent.
- status (VARCHAR): Status of the alert (e.g., Sent, Acknowledged, Dismissed).

Prediction

Description: Stores the results generated by machine learning models that predict extreme weather events that may affect specific locations.

Attributes:

- prediction_id (INT, PK): Unique identifier of the prediction.
- timestamp (DATETIME): Date and time the prediction was generated.
- event_type_id (INT, FK): Type of event predicted (e.g., drought, flood).
- probability (DECIMAL): Estimated probability of occurrence (%).
- severity_level_id (INT, FK): Estimated severity level of the event.
- model_id (INT, FK): ID of the model used to generate the prediction.

Event_type

Description: Defines the types of weather events that the system can predict.

Attributes:

- event_type_id (INT, PK): Unique identifier of the event type.
- evet_name (VARCHAR(100)): Name of the event (e.g., Flood, Heatwave).
- event_description (VARCHAR(200)): Description of the event type.

Severity_level

Description: Categorizes the severity level of a predicted event, helping to prioritize alerts and responses.

Attributes:

- severity_level_id (INT, PK): Unique identifier of the severity level.
- severity_type (VARCHAR(50)): Name of the severity level (e.g., Low, Medium, High).
- level_description (VARCHAR(200)): Description of the severity level.

Model

Description: Contains metadata about the predictive models used by AgroClima, including versioning and training information.

Attributes:

- model_id (INT, PK): Unique identifier of the model.
- model_name (VARCHAR(100)): Descriptive name of the model.
- version (DECIMAL): Version number of the model.
- training_timestamp (DATETIME): Date and time of the model's last training.

Weather_Station

Description: Represents the weather stations (physical or virtual) that collect meteorological data used by the system to feed predictions.

Attributes:

- station_id (INT, PK): Unique identifier of the station.
- name (VARCHAR(100)): Name or code of the weather station.
- location_id (INT, FK): Reference to the location where the station is situated.

Weather_Data

Description: Stores individual weather readings collected by a weather station at a specific timestamp. These readings are key inputs for generating predictions.

Attributes:

- data_id (INT, PK): Unique identifier of the data entry.
- station_id (INT, FK): Weather station that generated the data.
- timestamp (DATETIME): Date and time of the reading.
- temperature (FLOAT): Temperature reading (°C).
- humidity (FLOAT): Relative humidity (%).
- wind_speed (FLOAT): Wind speed (km/h or m/s).

Location

Description: Represents geographic locations associated with weather stations or prediction areas. Enables spatial organization of environmental data.

Attributes:

- location_id (INT, PK): Unique identifier of the location.
- name (VARCHAR(100)): Name of the location (e.g., city, region).
- latitude (DECIMAL): Latitude coordinate.
- longitude (DECIMAL): Longitude coordinate.
- type (VARCHAR(100)): Type of location (e.g., Station, Critical Zone, Rural Area).

Agriculture_field

Description: Represents a physical agricultural field owned or managed by a user. Each field is geolocated and can be associated with one or more crops. This entity allows AgroClima to tailor predictions and recommendations to specific land plots.

Attributes:

- field_id (INT, PK): Unique identifier of the agricultural field.
- user_id (INT, FK): Reference to the user (farmer) who owns or manages the field.
- field_type (INT): Type or classification of the field (e.g., irrigated, rainfed, greenhouse).

Crop

Description: Stores the crops cultivated in each agricultural field. This allows the platform to provide crop-specific recommendations and monitor weather impacts on individual crop types.

Attributes:

- crop_id (INT, PK): Unique identifier of the crop record.
- field_id (INT, FK): Field where the crop is grown.
- crop_name (VARCHAR(200)): Name of the crop (e.g., maize, rice, coffee).

Recommendation

Description: Stores system-generated recommendations based on predictions and field conditions. These are delivered to farmers to support agricultural decisions such as irrigation, fertilization, or harvesting.

Attributes:

- recommend_id (INT, PK): Unique identifier of the recommendation.
- recommend_descrip (VARCHAR(300)): Description of the recommendation.
- prediction_id (INT, FK): Reference to the prediction that triggered the recommendation.

Data_source

Description: Defines the external or internal sources of meteorological data used in the system. This allows tracking of data provenance and managing integration with third-party APIs.

Attributes:

- source_id (INT, PK): Unique identifier of the data source.
- name (VARCHAR(100)): Name of the source (e.g., OpenWeather, Copernicus).
- type (VARCHAR(50)): Type of source (e.g., API, physical station).
- url (VARCHAR(100)): URL or endpoint of the data source.

Weather_data_source

Description: This is an associative entity that links weather stations to their corresponding data sources. It allows tracking of when a station began using a specific data source.

Attributes:

- station_id (INT, FK): Reference to the weather station.
- source_id (INT, FK): Reference to the data source.
- start_date (DATETIME): Date when this data source started being used for the station.

References

European Commission (2020). *Copernicus Climate Data Store (CDS) User Guide*.

- Link: https://cds.climate.copernicus.eu