NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Data Science and Business Analytics"

Software Project Report on the Topic:

Title of the Project

Implementation of the Analytic Hierarchy Process in the PostgreSQL DBMS Environment

Fulfilled by:
Student of the Group БПАД212
Anikina Elizaveta

Assessed by the Project Supervisor:

Morgunov Evgenii
Senior Lecturer
Faculty of Computer Science, HSE University

Moscow 2024

Abstract

In English

Nowadays, in the rapidly developing digital world, decision-making is a complicated process that requires structural methods to effectively handle vast amounts of information. The Analytic Hierarchy Process (AHP) is a great tool for managing such complexity by breaking down difficult problems into more manageable sub-problems.

My project aims to improve decision-making capabilities within the PostgreSQL Database Management System by incorporating the Analytical Hierarchy Process (AHP) directly into its environment. This integration will allow users to solve complex decision-making tasks more efficiently, leveraging PostgreSQL's robust data management and computational features.

My project consists of several stages: studying AHP principles, designing a database schema, developing and creating custom AHP functions using PL/pgSQL, and conducting testing to ensure accuracy and reliability. I also developed a user-friendly web interface to simplify interaction with the AHP functionalities. The interface makes the project accessible to users with different levels of technical awareness.

The integration of AHP into PostgreSQL not only simplifies the decision-making process but also enhances the capabilities of PostgreSQL as a decision-support tool. This combination provides a powerful solution for organizations that have to make informed decisions based on wide data analysis.

In Russian

В современном мире цифровые технологии развиваются очень быстро и принятие решений – это сложный процессом, требующий структурированных методов для эффективной обработки огромных объемов информации. Метод Анализа Иерархий – это инструмент для управления такими сложными задачми путем разбиения их на более управляемые подзадачи.

Мой проект направлен на улучшение возможностей принятия решений в системе управления базами данных PostgreSQL путем интеграции Метода Анализа Иерархий непосредственно в среду СУБД. Эта интеграция позволит пользователям выполнять сложные задачи по принятию решений более эффективно, используя функции управления данными и вычислительные возможности PostgreSQL.

Мой проект состоит из нескольких этапов: изучение принципов Метода Анализа Иерархий, разработка схемы базы данных, разработка и создание пользовательских функций AHP с использованием PL/pgSQL и проведение тщательного тестирования для обеспечения точности и надежности. Я также разработала веб-интерфейс для упрощения взаимодействия с функциями Метода Анализа Иерархий, который делает их доступными для пользователей с различным уровнем технической подготовки.

Интеграция Метода Анализа Иерархий в PostgreSQL упрощает процесс принятия решений и раширяет возможности PostgreSQL как инструмента поддержки принятия решений. Это сочетание предоставляет решение для организаций, которым необходимо принимать решения на основе всестороннего анализа данных.

Introduction

Relevance

The Analytic Hierarchy Process (AHP) is a decision-making framework developed by Thomas L. Saaty in the 1970s. It stands out as a key tool for tackling complex tasks. The Analytic Hierarchy Process is aimed to simplify the decision-making process by dividing a complex problem into a hierarchy of smaller sub-problems. Each of these problems can be analyzed independently and then integrated into a cohesive whole. This methodology makes decision-making less complex and ensures that decisions are made based on a thorough analysis of all factors involved.

The relevance of AHP in modern decision-making cannot be overstated. It supports different applications, from strategic planning and resource allocation to selecting among competing alternatives. However, the practical implementation of AHP requires a robust computational platform that can handle complex calculations and data management tasks efficiently. PostgreSQL, an advanced open-source Database Management System (DBMS), emerges as an ideal platform for this purpose. PostgreSQL is known for its reliability and offers a conducive environment for implementing AHP. Its wide adoption across industries and ability to manage large datasets make it particularly suitable for integrating AHP into decision-making processes. By leveraging PostgreSQL, organizations can enhance their decision-making capabilities, making informed decisions that are critical for strategic success in today's competitive environment.

Goal

The primary goal of this project is to seamlessly integrate the Analytic Hierarchy Process (AHP) into the PostgreSQL Database Management System environment. This integration aims to enable PostgreSQL users to utilize AHP for complex decision-making processes directly within the database. By inserting AHP's systematic and hierarchical approach to evaluating alternatives based on multiple criteria, the project seeks to enhance the decision-making capabilities of PostgreSQL to make it a more powerful tool for data analysts, researchers, and businesses requiring sophisticated decision support systems.

Objectives

1. To review the fundamental principles of the Analytic Hierarchy Process (AHP) and assess the practicability of its integration into PostgreSQL, considering the database's architecture and features.
2. To design a schema and develop a functional prototype that integrates AHP into PostgreSQL, allowing for efficient data input, processing, and analysis within the DBMS environment.
3. To implement AHP algorithms as custom functions or stored procedures within PostgreSQL, ensuring accurate and efficient decision-making capabilities.
4. To conduct thorough testing of the AHP implementation in PostgreSQL, including performance benchmarking and optimization to handle large datasets and complex decision matrices.
5. To develop a user-friendly interface or set of tools that facilitate the interaction with the AHP functionality, ensuring ease of use for non-technical users.
6. To create comprehensive documentation that includes implementation details, user guides, and best practices, facilitating the adoption and effective use of the AHP functionality by the PostgreSQL community.
7. To evaluate the effectiveness of the AHP integration in real-world decision-making scenarios and integrate user feedback for continuous improvement.

Literature review

Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process was introduced by Thomas L. Saaty in 1980. It is a decision-making tool that decomposes complex decision problems into a hierarchical structure of goals, criteria, sub-criteria, and alternatives. Each level of the hierarchy is analyzed through pairwise comparisons, which are then synthesized to define overall priorities. Saaty's work[1] provides a comprehensive introduction to the methodology and its applications, highlighting its capability to transform subjective assessments into objective, quantifiable metrics.

Theoretical Foundations and Applications

Patrick T. Harker and Luis G. Vargas expanded on the theoretical foundations of AHP[2] by exploring the ratio scale estimation theory, which underpins the pairwise comparison method used in AHP. Their research delves into the mathematical robustness of AHP and its practical reliability in various decision-making contexts.

Fatemeh Zahedi conducted a survey of AHP's applications across different fields, demonstrating its high effectiveness. In her work[3] Zahedi emphasizes AHP's widespread adoption in areas such as supply chain management, healthcare, and environmental management, showcasing its practical utility in handling diverse and complex decision scenarios.

Integration of AHP into PostgreSQL

Integrating AHP into a Database Management System (DBMS) like PostgreSQL offers significant advantages by leveraging the DBMS's robust data management and computational capabilities. PostgreSQL is an advanced open-source object-relational database system that provides extensive support for custom functions and procedural languages such as PL/pgSQL. The official PostgreSQL documentation emphasizes the system's ability to handle complex queries and support for procedural languages, making it an ideal platform for implementing AHP's computational

[1] Saaty, T. L. (1980). The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. McGraw-Hill.
[2] Harker, P. T., & Vargas, L. G. (1990). The Theory of Ratio Scale Estimation: Saaty's Analytic Hierarchy Process. Management Science, 33(11), 1383-1403.
[3] Zahedi, F. (1986). The Analytic Hierarchy Process: A Survey of the Method and its Applications. Interfaces, 16(4), 96-108.

requirements. PostgreSQL's architecture facilitates efficient data processing and ensures scalability, which is crucial for complex decision-making frameworks like AHP.

Practical Implementations and User Interfaces

Adel Guitouni and Jean-Marc Martel in their work[4] provide practical guidelines for choosing appropriate multi-criteria decision analysis (MCDA) methods in their work *Tentative Guidelines to Help Choosing an Appropriate MCDA Method*. This study is particularly relevant for understanding the practical implementation of AHP in real-world scenarios, such as urban planning and strategic business decisions, and illustrates how AHP can support complex decision-making processes.

Theoretical part

The Analytic Hierarchy Process (AHP) is a multi-criteria decision-making decision-making tool that helps to evaluate a set of alternatives against a set of criteria. The methodology involves structuring the decision problem into a hierarchy, assessing the criteria and alternatives through pairwise comparisons, and calculating priority scales using eigenvalue methods. This process transforms subjective assessments into objective, quantifiable metrics, facilitating informed decision-making. AHP's mathematical foundation is based on matrices and eigenvectors. Decision-makers rate the importance of criteria and the preference of alternatives relative to each criterion. These ratings are used to construct comparison matrices. The principal eigenvector of each matrix represents the relative weights of criteria or the priority of alternatives. Consistency indices and ratios are calculated to evaluate the consistency of the decision-makers judgments, ensuring the reliability of the results.

PostgreSQL's architecture is known for its robustness and extensibility and provides a solid foundation for implementing AHP. It supports advanced data types, custom functions, and procedural languages like PL/pgSQL, which can be leveraged to implement AHP's computational requirements. The ability to create and execute stored procedures allows for the encapsulation of complex AHP calculations within the database, enabling efficient data processing and analysis. Furthermore, PostgreSQL's support for concurrent transactions and its comprehensive indexing mechanisms ensure that AHP implementations can scale effectively to accommodate large datasets and complex decision-making scenarios.

[4] Guitouni, A., & Martel, J. M. (1998). Tentative Guidelines to Help Choosing an Appropriate MCDA Method. European Journal of Operational Research, 109(2), 501-521.

The integration of AHP into the PostgreSQL environment capitalizes on AHP's structured approach to decision-making and PostgreSQL's advanced data management capabilities. This combination offers a powerful tool for addressing complex decision-making challenges, providing a systematic and quantifiable method for evaluating alternatives across a wide range of criteria.

Basic Terms and Definitions

Analytic Hierarchy Process (AHP): A decision-making framework that structures complex decisions into a hierarchy of simpler problems, using pairwise comparisons and mathematical models to derive priority scales.

PostgreSQL DBMS: An open-source, object-relational database management system known for its robustness, extensibility, and support for advanced data types and functionality.

Pairwise Comparison: A method used in AHP to compare elements (criteria or alternatives) in pairs to judge which element is preferable or has more importance, and to what extent.

Priority Vector: A vector that represents the relative priorities of elements (criteria or alternatives) derived from the pairwise comparison matrices in AHP.

Consistency Index (CI): A measure used in AHP to assess the consistency of the judgments made in the pairwise comparison process. It helps in evaluating the reliability of the decision-making process.

Eigenvalue: In the context of AHP, the principal eigenvalue of a comparison matrix is used to derive the priority vector and check the consistency of comparisons.

PL/pgSQL: A procedural language supported by PostgreSQL, allowing for the creation of stored procedures, functions, and triggers to encapsulate complex logic within the database.

Database Schema: The structure of a database defined in a formal language supported by the DBMS. It describes the tables, fields, relationships, views, indexes, and other elements.

Stored Procedure: A subroutine available to applications accessing a relational database system. It is stored in the database and can be executed by the database engine to perform complex operations.

Decision Matrix: A tool used in decision analysis to evaluate and prioritize a list of alternatives based on specific criteria, often utilized within the AHP framework.

Chapter 1

Designing the Database Schema

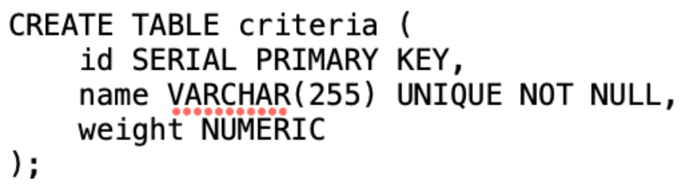
Designing a database schema is a critical step in integrating the Analytic Hierarchy Process (AHP) into PostgreSQL. The schema defines the structure of the database, outlining how data is organized, stored, and managed. For my project, the schema needs to support the hierarchical nature of AHP and facilitate the efficient calculation of priority vectors and consistency ratios.

The main objectives in designing the database schema are:

1. Define tables to store criteria, alternatives, and pairwise comparisons.
2. Ensure data integrity and consistency through constraints and relationships.
3. Optimize for efficient data retrieval and processing.

Criteria Table

The **criteria** table stores the decision criteria used in the AHP process. Each criterion is assigned a unique identifier, a name, and a weight. The weight will store the calculated importance of each criterion after the AHP calculations are performed

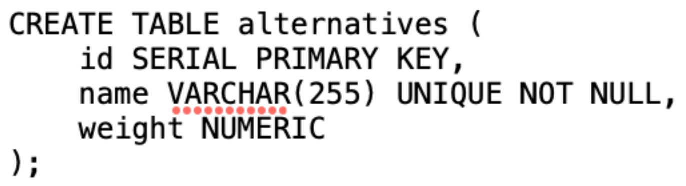


```
CREATE TABLE criteria (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE NOT NULL,
    weight NUMERIC
);
```

- **id**: A unique identifier for each criterion.
- **name**: The name of the criterion.
- **weight**: The calculated weight of the criterion.

Alternatives Table

The **alternatives** table stores the different alternatives being evaluated. Similar to the criteria table, each alternative has a unique identifier, a name, and a weight. The weight reflects the calculated preference of each alternative relative to the criteria. This structure facilitates the comparison and evaluation of different options.



```
CREATE TABLE alternatives (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE NOT NULL,
    weight NUMERIC
);
```

- **id**: A unique identifier for each alternative.
- **name**: The name of the alternative.
- **weight**: The calculated weight of the alternative.

Criteria Comparisons Table

The **criteria_comparisons** table stores pairwise comparisons between criteria. These comparisons are used to determine the relative importance of each criterion. Each entry in this table includes two criteria being compared and the value of the comparison, indicating how much more important one criterion is compared to the other. This data forms the basis for calculating the priority vectors of the criteria.

```
CREATE TABLE criteria_comparisons (
    id SERIAL PRIMARY KEY,
    criteria1_id INT REFERENCES criteria(id),
    criteria2_id INT REFERENCES criteria(id),
    value NUMERIC
);
```

- **id**: A unique identifier for each comparison.
- **criteria1_id**: The first criterion in the pairwise comparison.
- **criteria2_id**: The second criterion in the pairwise comparison.
- **value**: The comparison value, indicating how much more important **criteria1_id** is compared to **criteria2_id**.

Alternative Comparisons Table

The **alternative_comparisons** table stores pairwise comparisons between alternatives for each criterion. This data is used to determine the relative preference for each alternative with respect to each criterion. Each entry includes two alternatives being compared, the criterion under which the comparison is made, and the value of the comparison. This structure allows for a detailed evaluation of how each alternative performs against each criterion.

```
CREATE TABLE alternative_comparisons (
    id SERIAL PRIMARY KEY,
    alternative1_id INT REFERENCES alternatives(id),
    alternative2_id INT REFERENCES alternatives(id),
    criteria_id INT REFERENCES criteria(id),
    value NUMERIC
);
```

- **id**: A unique identifier for each comparison.
- **alternative1_id**: The first alternative in the pairwise comparison.
- **alternative2_id**: The second alternative in the pairwise comparison.
- **criteria_id**: The criterion under which the comparison is made.
- **value**: The comparison value, indicating how much more preferred **alternative1_id** is compared to **alternative2_id** for the given criterion.

Developing Custom AHP Functions

To facilitate the AHP calculations, I implemented several functions using PL/pgSQL. These functions perform the necessary mathematical operations to derive priority vectors and consistency ratios from the pairwise comparisons stored in the tables.

Priority Vector

The function to calculate the priority vector from a pairwise comparison matrix is a fundamental part of the Analytic Hierarchy Process (AHP). The priority vector represents the relative weights of the elements (criteria or alternatives) being compared. This function involves several key steps, including initializing necessary variables, calculating column sums, normalizing the comparison matrix, and averaging the normalized values to derive the priority vector.

Methodology

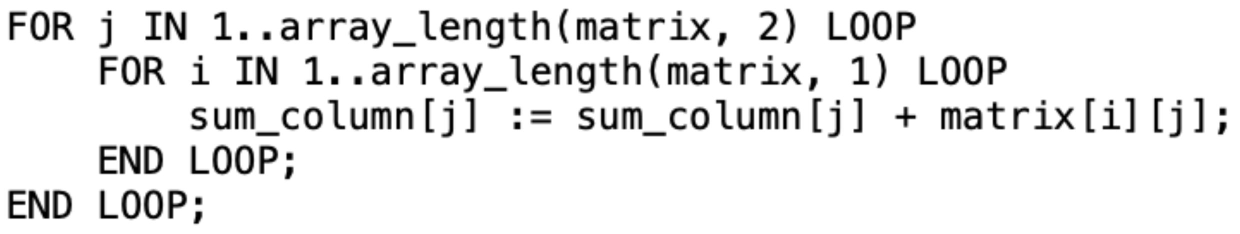
1. Initialization

   The function begins by initializing an array to store the sum of each column in the comparison matrix. This array is essential for normalizing the matrix.

```
sum_column := ARRAY(SELECT 0 FROM generate_series(1, array_length(matrix, 1)));
```

2. Calculate Column Sums

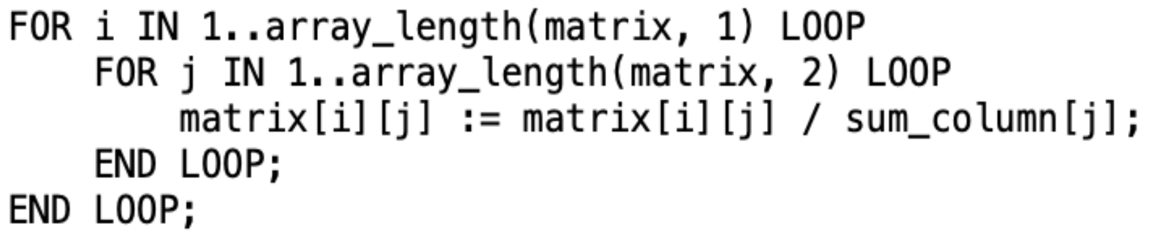
   The function then iterates through each element of the matrix to calculate the sum of each column. This step ensures that each element in the matrix is appropriately weighted relative to others in the same column.



```
FOR j IN 1..array_length(matrix, 2) LOOP
    FOR i IN 1..array_length(matrix, 1) LOOP
        sum_column[j] := sum_column[j] + matrix[i][j];
    END LOOP;
END LOOP;
```

3. Normalize the Matrix

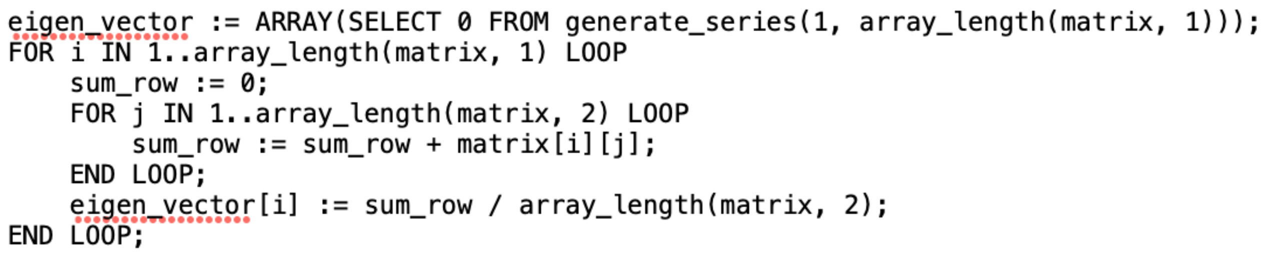
   The next step is to normalize the comparison matrix. Normalization involves dividing each element in the matrix by the sum of its respective columns. This step converts the comparison values into a consistent scale, allowing for meaningful comparisons across different elements.



```
FOR i IN 1..array_length(matrix, 1) LOOP
    FOR j IN 1..array_length(matrix, 2) LOOP
        matrix[i][j] := matrix[i][j] / sum_column[j];
    END LOOP;
END LOOP;
```

4. Calculate the Priority Vector

   The priority vector is calculated by averaging the normalized values in each row of the matrix. This average represents the relative weight of each element. The function iterates through each row, sums the normalized values, and divides by the number of elements to get the average.



```
eigen_vector := ARRAY(SELECT 0 FROM generate_series(1, array_length(matrix, 1)));
FOR i IN 1..array_length(matrix, 1) LOOP
    sum_row := 0;
    FOR j IN 1..array_length(matrix, 2) LOOP
        sum_row := sum_row + matrix[i][j];
    END LOOP;
    eigen_vector[i] := sum_row / array_length(matrix, 2);
END LOOP;
```

Benefits

- The normalization step ensures that all comparisons are on a consistent scale, which enhances the reliability of the results. This consistency is crucial for accurate decision-making.
- By performing these calculations within the database, the function leverages PostgreSQL's robust computational capabilities. This approach leads to faster processing times compared to external scripts, significantly improving efficiency.

- This method is highly scalable and capable of handling large matrices efficiently. This scalability makes it suitable for complex decision-making scenarios involving many criteria and alternatives, ensuring that the system can grow with the needs of the users.

Reshape an Array

The 'array_reshape' function is very important within the Analytic Hierarchy Process (AHP) implementation in PostgreSQL. It converts a one-dimensional array into a two-dimensional matrix. This transformation is essential for preparing the linear array of comparison values into a format that can be used for matrix operations required in AHP calculations.

This function reshapes a one-dimensional array into a two-dimensional matrix. This is useful for converting the linear array of comparison values into a matrix format that can be used for further calculations. It ensures that the data is correctly formatted for matrix operations.

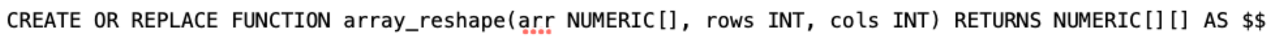
Methodology

1. Function Declaration and Parameter Definition

   The function 'array_reshape' is declared to accept three parameters:

   - **'arr'**: The one-dimensional array of numeric values that needs to be reshaped.
   - **'rows'**: The number of rows the resulting two-dimensional matrix should have.
   - **'cols'**: The number of columns the resulting two-dimensional matrix should have.

   The function is defined to return a two-dimensional array of numeric values.

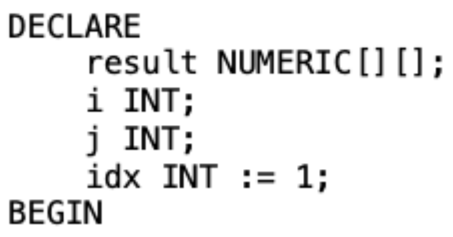


```
CREATE OR REPLACE FUNCTION array_reshape(arr NUMERIC[], rows INT, cols INT) RETURNS NUMERIC[][] AS $$
```

2. Variable Declaration

   Within the function, I declared several variables:

   - **'result'**: A two-dimensional array that will store the reshaped matrix.
   - **'i', 'j'**: Loop counters used to iterate over the rows and columns of the matrix.
   - **'idx'**: An index variable initialized to 1, which is used to traverse the one-dimensional array.



```
DECLARE
    result NUMERIC[][];
    i INT;
    j INT;
    idx INT := 1;
BEGIN
```

3. Initialize the Result Matrix

The 'result' matrix is initialized with the specified number of rows and columns. Each element in the matrix is initially set to zero.

```
    result := ARRAY(SELECT ARRAY(SELECT 0 FROM generate_series(1, cols)) FROM generate_series(1,
rows));
```

Here, 'generate_series(1, cols)' creates an array of zeros with a length equal to the number of columns, and 'generate_series(1, rows)' repeats this array for the number of rows.

4. Populate the Result Matrix

   The function populates the 'result' matrix with values from the one-dimensional array 'arr'. It uses loops to iterate over each element in the 'result' matrix. The outer loop runs from 1 to the number of rows. The inner loop runs from 1 to the number of columns.

   Within the inner loop, the value at the current index of 'arr' is assigned to the current position in the 'result' matrix. The index 'idx' is incremented after each assignment to move to the next element in 'arr'.

```
FOR i IN 1..rows LOOP
    FOR j IN 1..cols LOOP
        result[i][j] := arr[idx];
        idx := idx + 1;
    END LOOP;
END LOOP;
```

Benefits

- It ensures that linear data can be formatted into matrices, which is essential for performing complex calculations.
- This function provides flexibility by allowing dynamic reshaping of arrays, accommodating various matrix sizes needed at different stages of AHP.
- It simplifies the process of converting one-dimensional arrays into two-dimensional matrices directly within the database, thereby eliminating the need for external processing.

Criteria Weights

The 'calculate_criteria_weights' function is designed to compute and update the weights of the criteria based on pairwise comparisons. This process ensures that the criteria weights accurately reflect their relative importance as determined by the pairwise comparison matrix.

The primary goal of this function is to:

1. Retrieve the pairwise comparison matrix for the criteria.
2. Reshape the one-dimensional array of comparison values into a two-dimensional matrix.
3. Calculate the priority vector (eigenvector) from the matrix.
4. Update the weights of the criteria in the database with the calculated priority vector.

Methodology

1. Function Declaration

   The function is declared without any input parameters and returns 'VOID'.

```
CREATE OR REPLACE FUNCTION calculate_criteria_weights() RETURNS VOID AS $$
```

2. Variable Declaration

   Variables declared within the function:

   - **'matrix'**: A two-dimensional numeric array that will store the pairwise comparison matrix.
   - **'eigen_vector'**: A numeric array that will hold the priority vector calculated from the matrix.
   - **'criteria_count'**: An integer to store the number of criteria.

```
DECLARE
    matrix NUMERIC[][];
    eigen_vector NUMERIC[];
    criteria_count INT;
BEGIN
```

3. Retrieve the Pairwise Comparison Matrix

   The function starts by counting the number of criteria and fetching the pairwise comparison matrix from the 'criteria_comparisons' table. The 'array_agg' function is used to aggregate the comparison values into a one-dimensional array ordered by 'criteria1_id' and 'criteria2_id'.

```
criteria_count := (SELECT count(*) FROM criteria);

SELECT array_agg(value ORDER BY criteria1_id, criteria2_id) INTO matrix
FROM criteria_comparisons;
```

Benefits

- It ensures accuracy by calculating criteria weights based on systematic pairwise comparisons. This method guarantees that the relative importance of each criterion is determined objectively.
- The function automates the update of criteria weights, thereby reducing the need for manual adjustments and minimizing potential errors associated with manual updates.

Alternative Weights

This function is designed to compute and update the weights of alternatives based on pairwise comparisons for each criterion. This function processes the pairwise comparison matrices for each criterion, calculates the priority vectors for the alternatives, and combines these vectors with the criteria weights to determine the overall weights of the alternatives. Finally, it updates the weights of the alternatives in the database.

Methodology

1. Function Declaration

```
CREATE OR REPLACE FUNCTION calculate_alternative_weights() RETURNS VOID AS $$
```

   The function is declared without any input parameters and returns 'VOID'.

2. Variable declaration

   Variables declared within the function:

   - **'criteria_row'**: A record to iterate over each criterion.
   - **'matrix'**: A two-dimensional numeric array that will store the pairwise comparison matrix for alternatives.
   - **'eigen_vector'**: A numeric array that will hold the priority vector calculated from the matrix.
   - **'criteria_weight'**: A numeric value to store the weight of the current criterion.
   - **'alternative_weights'**: A numeric array to accumulate the weighted priorities of the alternatives.
   - **'i', 'j'**: Loop counters used to iterate over the rows and columns of the matrix.
   - **'alternatives_count'**: An integer to store the number of alternatives.

```
DECLARE
    criteria_row RECORD;
    matrix NUMERIC[][];
    eigen_vector NUMERIC[];
    criteria_weight NUMERIC;
    alternative_weights NUMERIC[] := ARRAY(SELECT 0 FROM generate_series(1, (SELECT count(*) FROM
alternatives)));
    i INT;
    j INT;
    alternatives_count INT;
BEGIN
```

3. Count the Number of Alternatives

   The function starts by counting the number of alternatives.

```
alternatives_count := (SELECT count(*) FROM alternatives);
```

4. Iterate Over Each Criterion

   The function iterates over each criterion to process the pairwise comparison matrices for the alternatives.

```
FOR criteria_row IN SELECT id, weight FROM criteria LOOP
```

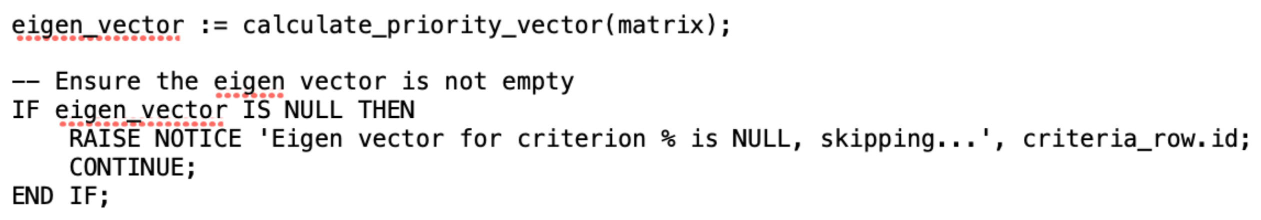
5. Fetch and Reshape the Pairwise Comparison Matrix

   Within the loop, the function fetches the pairwise comparison matrix for the alternatives under the current criterion. The 'array_agg' function is used to aggregate the comparison values into a one-dimensional array ordered by 'alternative1_id' and 'alternative2_id'. The function then checks if the size of the array matches the expected size and reshapes it into a two-dimensional matrix using the 'array_reshape' function.

```
        SELECT array_agg(value ORDER BY alternative1_id, alternative2_id) INTO matrix
        FROM alternative_comparisons
        WHERE criteria_id = criteria_row.id;

        -- Check if the matrix is correctly formed
        IF array_length(matrix, 1) IS DISTINCT FROM alternatives_count * alternatives_count THEN
            RAISE NOTICE 'Matrix size mismatch for criterion %: expected %, got %', criteria_row.id,
alternatives_count * alternatives_count, array_length(matrix, 1);
            CONTINUE;
        END IF;

        -- Reshape the one-dimensional array into a two-dimensional matrix
        matrix := array_reshape(matrix, alternatives_count, alternatives_count);

        -- Ensure the matrix is not empty
        IF matrix IS NULL THEN
            RAISE NOTICE 'Matrix for criterion % is NULL, skipping...', criteria_row.id;
            CONTINUE;
        END IF;
```

6. Calculate the Priority Vector

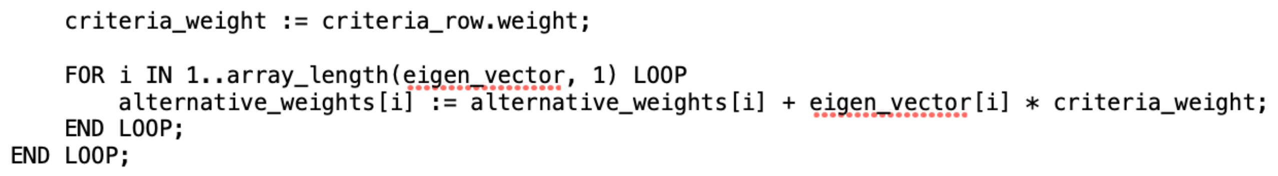
   The function calculates the priority vector (eigenvector) for the alternatives using the 'calculate_priority_vector' function. If the resulting eigenvector is NULL, a notice is raised, and the function continues to the next criterion.



```
eigen_vector := calculate_priority_vector(matrix);

-- Ensure the eigen vector is not empty
IF eigen_vector IS NULL THEN
    RAISE NOTICE 'Eigen vector for criterion % is NULL, skipping...', criteria_row.id;
    CONTINUE;
END IF;
```

7. Combine Priority Vectors with Criteria Weights

   The function multiplies each element of the priority vector by the weight of the current criterion ('criteria_weight'). This weighted priority is accumulated in the 'alternative_weights' array.



```
    criteria_weight := criteria_row.weight;

    FOR i IN 1..array_length(eigen_vector, 1) LOOP
        alternative_weights[i] := alternative_weights[i] + eigen_vector[i] * criteria_weight;
    END LOOP;
END LOOP;
```

Benefits

- It ensures accuracy by calculating alternative weights based on systematic pairwise comparisons for each criterion, reflecting their true relative importance.
- The function automates the update of alternative weights, reducing the need for manual adjustments and minimizing the potential for errors associated with manual processes.

Chapter 2

Web Interface

To enhance the usability and accessibility of the Analytic Hierarchy Process (AHP) integrated into PostgreSQL, developing a web interface is a crucial step. The web interface will provide users with a user-friendly platform to interact with the AHP system, enabling them to input data, perform analyses, and visualize results without requiring advanced technical skills.

Objectives

The primary objectives of the web interface are to:

1. Simplify the data input process for criteria, alternatives, and pairwise comparisons.
2. Provide an intuitive platform for users to perform AHP analyses.
3. Display results in a clear and understandable format.
4. Ensure accessibility for users with varying levels of technical expertise.

Technologies Used

The web interface was developed using the following technologies to ensure a responsive and interactive user experience:

- Python: The primary programming language for backend development.
- Flask: A lightweight web framework for building the web application.
- HTML5: For structuring the web pages.
- SQLAlchemy: For Object Relational Mapping (ORM) to interact with PostgreSQL.
- PostgreSQL: The database where all the AHP data and calculations are stored.

Features

Data Input Forms

Users can add new criteria by entering the name of each criterion in a dedicated form. The form ensures that each criterion is uniquely identified and properly recorded in the database. This step is crucial as it sets the foundation for subsequent pairwise comparisons and weight calculations.

Users can add new alternatives by entering the name of each alternative in a separate form. This form ensures that each alternative is distinctly identified and stored in the database, ready for comparison against the defined criteria.

The forms include validation checks to ensure that the entered data meets the required standards. For example, fields cannot be left blank, and duplicate entries are prevented. This validation helps maintain the integrity and accuracy of the data.

Pairwise Comparison

The interface presents pairs of criteria or alternatives to the user one at a time, prompting them to indicate which item in each pair is preferred and to what extent. This process is done using a scale, where users can express their preference intensity.

As users make their comparisons, the interface dynamically updates the pairwise comparison matrices stored in the database. This ensures that all data is captured in real-time and is immediately available for subsequent analysis.

Calculation and Analysis

Once data is entered, users can trigger the calculation process, and the backend handles the execution of predefined functions to compute the weights of criteria and alternatives.

Chapter 3

Examples

To illustrate the method and demonstrate the practical application of integrating the Analytic Hierarchy Process (AHP) into PostgreSQL, I have developed three comprehensive examples. These examples showcase how this integration can enhance decision-making capabilities across different scenarios. Each example involves defining criteria and alternatives, performing pairwise comparisons, calculating weights, and updating the database with the results. Through these examples, I wanted to demonstrate the practical utility and effectiveness of the proposed method and its potential applications in real-world decision-making processes.

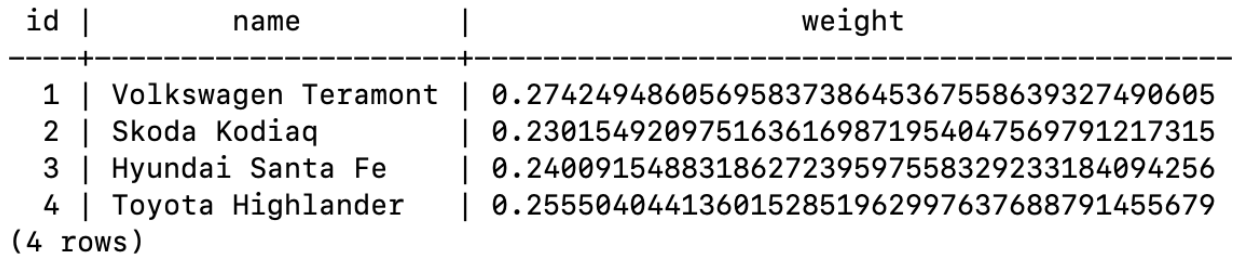
Cars

For the first example, I chose to compare four family car options to demonstrate the practical application of the AHP integration. I began by identifying the most important criteria for evaluating a family car: Price, Safety, Comfort, Efficiency, and Reliability. These criteria are crucial in determining the overall suitability of a car for family use.

Next, I selected four popular family car models based on my research: Volkswagen Teramont, Skoda Kodiaq, Hyundai Santa Fe, and Toyota Highlander. These models represent a range of choices commonly considered by families.

I then inserted the criteria and alternatives into the pre-established tables in the PostgreSQL database. Utilizing the previously defined functions, I calculated the weights of both the criteria and the alternatives. The criteria weights indicate their relative importance, while the alternatives' weights reflect their relative preference for each criterion.

The analysis revealed that the Volkswagen Teramont is the best car for family use among the options, with an alternative weight of 0.27. This weight is significantly higher—by 0.02—than the next best option, highlighting the Volkswagen Teramont's superior suitability based on the selected criteria.



```
 id |         name         |                  weight
----+----------------------+------------------------------------------
  1 | Volkswagen Teramont | 0.2742494860569583738645367558639327490605
  2 | Skoda Kodiaq        | 0.2301549209751636169871954047569791217315
  3 | Hyundai Santa Fe    | 0.2400915488318627239597558329233184094256
  4 | Toyota Highlander   | 0.2555040441360152851962997637688791455679
(4 rows)
```

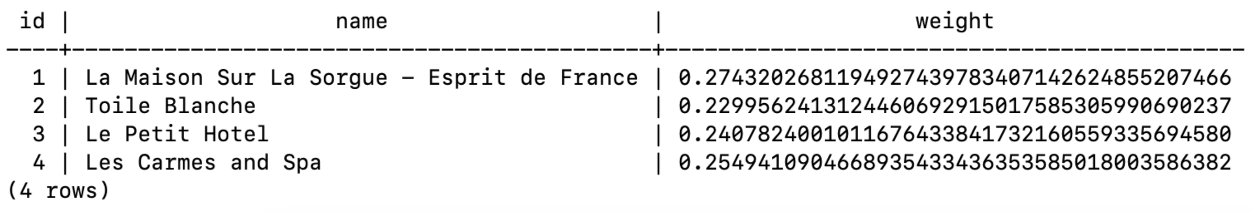
Hotels

For the second example, I chose to compare four hotel options to demonstrate the practical application of the AHP integration. I began by identifying the most important criteria for evaluating a hotel: Price, Safety, Comfort, Location, and Amenities. These criteria are crucial in determining the overall suitability of a hotel for a vacation.

Next, I selected four popular hotels based on my research: La Maison Sur La Sorgue - Esprit de France, Toile Blanche, Le Petit Hotel, and Les Carmes and Spa. These hotels represent a range of choices commonly considered by travelers.

I then inserted the criteria and alternatives into the pre-established tables in the PostgreSQL database. Utilizing the previously defined functions, I calculated the weights of both the criteria and the alternatives. The criteria weights indicate their relative importance, while the alternatives' weights reflect their relative preference for each criterion.

The analysis revealed that Les Carmes and Spa is the best hotel for a vacation among the options.



```
 id |                    name                    |                  weight
----+--------------------------------------------+------------------------------------------
  1 | La Maison Sur La Sorgue - Esprit de France | 0.2743202681194927439783407142624855207466
  2 | Toile Blanche                              | 0.2299562413124460692915017585305990690237
  3 | Le Petit Hotel                             | 0.2407824001011676433841732160559335694580
  4 | Les Carmes and Spa                         | 0.2549410904668935433436353585018003586382
(4 rows)
```

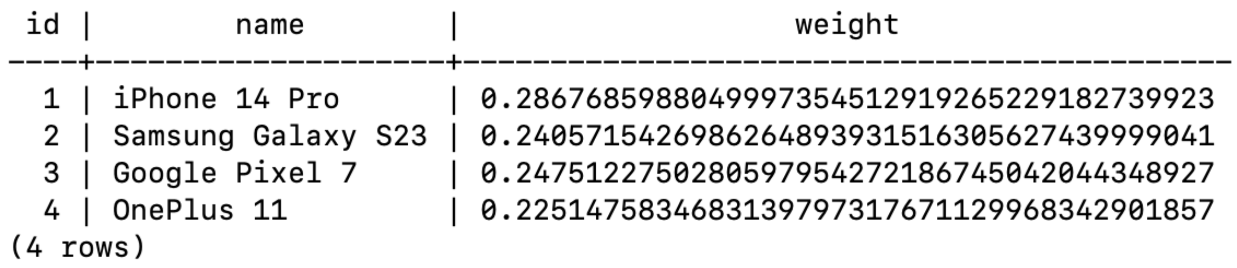
Phones

For the third example, I chose to compare four smartphone options to demonstrate the practical application of the AHP integration. I began by identifying the most important criteria for evaluating a smartphone: Price, Performance, Camera, Battery Life, and Design. These criteria are crucial in determining the overall suitability of a smartphone for a wide range of users.

Next, I selected four popular smartphone models based on my research: iPhone 14 Pro, Samsung Galaxy S23, Google Pixel 7, and OnePlus 11. These models represent a variety of choices commonly considered by consumers.

I then inserted the criteria and alternatives into the pre-established tables in the PostgreSQL database. Utilizing the previously defined functions, I calculated the weights of both the criteria and the alternatives. The criteria weights indicate their relative importance, while the alternatives' weights reflect their relative preference for each criterion.

The analysis revealed that the iPhone 14 Pro is the best smartphone among the options, with an alternative weight of 0.287. This weight is significantly higher—by 0.04—than the next best option, highlighting the iPhone 14 Pro's superior suitability based on the selected criteria.



```
 id |        name        |                  weight
----+--------------------+------------------------------------------
  1 | iPhone 14 Pro      | 0.2867685988049997354512919265229182739923
  2 | Samsung Galaxy S23 | 0.2405715426986264893931516305627439999041
  3 | Google Pixel 7     | 0.2475122750280597954272186745042044348927
  4 | OnePlus 11         | 0.2251475834683139797317671129968342901857
(4 rows)
```

Conclusion and Results

The integration of the Analytic Hierarchy Process (AHP) into the PostgreSQL Database Management System has been a successful initiative that significantly enhances decision-making capabilities. By incorporating AHP's systematic approach to evaluating alternatives based on multiple criteria directly within PostgreSQL, this project demonstrates the feasibility and effectiveness of combining robust data management with decision-support methodologies.

The development process involved several critical stages, including designing a comprehensive database schema, creating custom AHP functions using PL/pgSQL, and developing a user-friendly web interface using Flask. Each stage was crucial in ensuring that the system could efficiently handle complex decision-making tasks while remaining accessible to users with varying levels of technical expertise.

The database schema was carefully crafted to support the hierarchical structure of AHP, facilitating efficient storage and retrieval of criteria, alternatives, and pairwise comparisons. The custom PL/pgSQL functions enabled precise calculations of priority vectors and consistency ratios, leveraging PostgreSQL's computational capabilities for optimal performance.

The implementation of the web interface further simplified user interactions, allowing for seamless data input, analysis, and visualization of results. This interface, built using Python, Flask, SQLAlchemy, and HTML5, ensured that users could perform AHP analyses without extensive technical knowledge, thus broadening the tool's accessibility.

Overall, the project successfully demonstrates how integrating AHP into PostgreSQL can provide a powerful decision-support tool capable of handling complex and large-scale decision-making scenarios. The combination of AHP's methodological rigor and PostgreSQL's data management strengths offers a comprehensive solution for organizations aiming to make informed, data-driven decisions.

The results of this project are illustrated through three practical examples: comparing family cars, evaluating hotels, and assessing smartphones. These examples highlight the practical application of the AHP integration and its impact on decision-making processes.

List of Resources

- Saaty, T. L. (1980). The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. McGraw-Hill.
- Munier, N., & Hontoria, E. (2021). Uses and Limitations of the AHP Method. In *Management for professionals*. https://doi.org/10.1007/978-3-030-60392-2
- Harker, P. T., & Vargas, L. G. (1990). The Theory of Ratio Scale Estimation: Saaty's Analytic Hierarchy Process. Management Science, 33(11), 1383-1403.
- Zahedi, F. (1986). The Analytic Hierarchy Process: A Survey of the Method and its Applications. Interfaces, 16(4), 96-108.
- Guitouni, A., & Martel, J. M. (1998). Tentative Guidelines to Help Choosing an Appropriate MCDA Method. European Journal of Operational Research, 109(2), 501-521.
- Dey, P. K., & Ramcharan, E. K. (2008). Analytic Hierarchy Process Helps Select Site for Limestone Quarry Expansion in Barbados. Journal of Environmental Management, 88(4), 1381-1390.
- PostgrePro Education. Available at: https://postgrespro.ru/education
- *MindTools | Home*. (n.d.). https://www.mindtools.com/a7y139c/the-analytic-hierarchy-process-ahp
- Моргунов, Е. П., Моргунова, О. Н., Министерство образования и науки Российской Федерации, Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева, & редакционно-издательский совет университета. (2018). *Технологии разработки программ в среде операционных систем Linux и FreeBSD. Вводный курс* (p. 207) [Учеб. пособие]. Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. http://www.morgunov.org/docs/free_soft_tech_2.pdf
- *Моргуновы Е. П. и О. Н.: преподавание*. (n.d.). http://www.morgunov.org/teaching.html