PostgreSQL User Data Extraction and Loading

# Overview

This project demonstrates how to extract user information from the `/etc/passwd` file on a Linux system, transform the data, and load it into a PostgreSQL table using a shell script.

# Table of Contents

- [Overview](#overview)  
- [Pre-requisites](#pre-requisites)  
- [Steps](#steps)  
 - [1. Create PostgreSQL Table](#1-create-postgresql-table)  
 - [2. Create Shell Script](#2-create-shell-script)  
 - [3. Extract Data](#3-extract-data)  
 - [4. Transform Data](#4-transform-data)  
 - [5. Load Data to PostgreSQL](#5-load-data-to-postgresql)  
 - [6. Verify Data Load](#6-verify-data-load)  
- [Running the Script](#running-the-script)  
- [Conclusion](#conclusion)

# Pre-requisites

Before you begin, ensure you have the following:

- A working PostgreSQL instance.  
- Access to the `template1` PostgreSQL database.  
- Linux-based system (or any system with access to `/etc/passwd`).  
- Shell scripting knowledge.

# Steps

## 1. Create PostgreSQL Table

Start by connecting to the PostgreSQL database (`template1`) and creating a table to hold the user account information.

\c template1

Then, create the `users` table:

CREATE TABLE users (  
 username VARCHAR(50),  
 userid INT,  
 homedirectory VARCHAR(100)  
);

## 2. Create Shell Script

Create a new shell script file named `csv2db.sh` in your terminal:

touch csv2db.sh

Open the file in an editor and add the following header:

# This script extracts data from the /etc/passwd file into a CSV file.  
# It then loads the data into the PostgreSQL 'users' table.

## 3. Extract Data

To extract user data from the `/etc/passwd` file, we use the `cut` command. This extracts the username, user ID, and home directory.

# Extract phase

echo "Extracting data"  
cut -d":" -f1,3,6 /etc/passwd > extracted-data.txt

## 4. Transform Data

After extraction, the data is separated by colons. We need to convert it into CSV format by replacing the colons with commas.

# Transform phase

echo "Transforming data"  
tr ":" "," < extracted-data.txt > transformed-data.csv

## 5. Load Data to PostgreSQL

Now, we load the transformed CSV data into the PostgreSQL table using the `COPY` command.

# Load phase

echo "Loading data"  
export PGPASSWORD=<yourpassword>;

echo "\c template1;\COPY users FROM '/path/to/transformed-data.csv' DELIMITER ',' CSV;" | psql --username=postgres --host=localhost template1

Make sure to replace `<yourpassword>` with your actual PostgreSQL password and adjust the file path to the correct location of `transformed-data.csv`.

## 6. Verify Data Load

To ensure that the data is loaded correctly, we will run a query to fetch all records from the `users` table.

echo "SELECT \* FROM users;" | psql --username=postgres --host=localhost template1

This command will display all the rows loaded into the `users` table.

# Running the Script

1. Make the script executable:

chmod +x csv2db.sh

2. Run the script:

bash csv2db.sh

3. Verify that the data has been successfully loaded into the PostgreSQL database:

psql --username=postgres --host=localhost template1 -c "SELECT \* FROM users;"

# Conclusion

### ETL Process in This Project

This project demonstrates a basic **Extract, Transform, Load (ETL)** process using shell scripting and PostgreSQL:

1. **Extract**: The script extracts data from the system's /etc/passwd file. Specifically, it retrieves the username, user ID, and home directory fields using the cut command. This data represents the user information available on the system.
2. **Transform**: After extracting the data, it is originally in a colon (:) delimited format. The script then transforms this data into a comma-separated format (CSV) using the tr command, which replaces colons with commas. This step makes the data ready for loading into a relational database.
3. **Load**: The transformed CSV data is then loaded into a PostgreSQL database table (users). The COPY command is used within the script to insert the CSV data directly into the database, ensuring an efficient bulk load operation.