Politechnika Warszawska
Wydział Elektryczny

Uladzislau Hubar, Oleksandr Kolomiiets,
Yanina Shchaslyva, Lizaveta Niamera, Jan Konarski

„Hurtownie Danych. Projekt Bitcoin Blockchain ETL”

Sprawdzający:
mgr inż. Krzysztof Marek

Warszawa 2021

Spis treści

Opis projektu

1.1 Cel projektu

Celem projektu jest zaimplementowanie procesu „Extract, Transform and Load" (ETL) dla danych z Blockchain'u kryptowaluty Bitcoin. Jako efekt końcowy, będzie przedstawiony system, umożliwiający automatyczne pobieranie danych z Blockchain'u, klasteryzację danych oraz zamieszczenie przetransformowanych danych w bazie danych. Przechowywanie przetworzonych danych w bazie umożliwi łatwe pobieranie potrzebnych danych do dalniejszych analiz.

W celu ułatwienia zadania, skorzystamy z ogólnodostępnego opisu realizacji procesu ETL dla Blockchain'a [1], natomiast bardziej się skupimy na praktycznej realizacji.

1.2 Wstęp teoretyczny

Pobieranie (Extract):
Dane wejściowe będą pobierane z serwisu **Blokchain.info**.

W celu łatwiejszego przetwarzania danych, pobierane będą 150 bloków informacji naraz (informacja o transakcjach za 1 dzień).

Transformacja (Transformation):
Proces transformacji danych polega na zastosowaniu algorytmu Klasteryzacji, którego celem będzie zgrupowanie danych o transakcjach dla jednakowych adresów. Algorytm klasteryzacji składa z kilku różnych algorytmów:

1. Algorytmu wyszukiwania transakcji
2. Algorytmu wyszukiwania adresów
3. Algorytmu aktualizacji

Algorytm klasteryzacji (Clustering algorithm)

Algorithm 1: Clustering(inputSection)

```
Input: inputSection
Output: A list ec of clustered entities
index ← index max between entities;
i ← 0;
if index ≠ none then
    i ← 1;
else
end
i ← index + 1;
while (there are data to explore) do
    tr ← transaction list;
    exTr ← transaction explored;
    addr ← addresses list;
    exAddr ← addresses explored;
    while (list of address is not empty) do
        findTransactionsInexplored(addr, tr, exAddr);
        findAddressesInexplored(addr, tr, exTr);
    end
    deleteRawExplored();
end
updateEntities(exAddr);
ec ←updateTable(i, exAddr);
return ec;
```

Rysunek 1.1: Pseudokod algorytmu klasteryzacji [1]

Algorytm wyszukiwania transakcji (Transaction search algorithm)

Algorithm 2: findTransactionsInexplored(addr, tr, exAddr)

```
Input: A list addr of addresses to explore,
       A list tr of transactions to explore,
       A list exAddr of explored addresses
Output: A list tr of transactions inexplored
if (there are addresses in addr) then
    firstA ← first address of the list;
    find all transactions in which it exist by excluding the ones that are already explored;
    add the transactions found in the queue of tr;
    add firstA in exAddr;
    remove firstA from addr;
else
end
return tr;
```

Rysunek 1.2: Pseudokod algorytmu wyszukiwania transakcji [1]

Algorytm wyszukiwania adresów (Adress search algorithm)

Algorithm 3: findAddressesInexplored(addr, tr, exTr)

```
Input: A list addr of addresses to explore,
       A list tr of transactions to explore,
       A list exTr of explored transactions
Output: A list add of addresses inexplored
if (there are transactions to be explored) then
    firstT ← first transaction of the list;
    find all addresses in which it exist by excluding the ones that are already explored;
    add addresses found in the queue of addr;
    add firstT in exTr;
    remove firstT from tr;
else
end
return addr;
```

Rysunek 1.3: Pseudokod algorytmu wyszukiwania adresów [1]

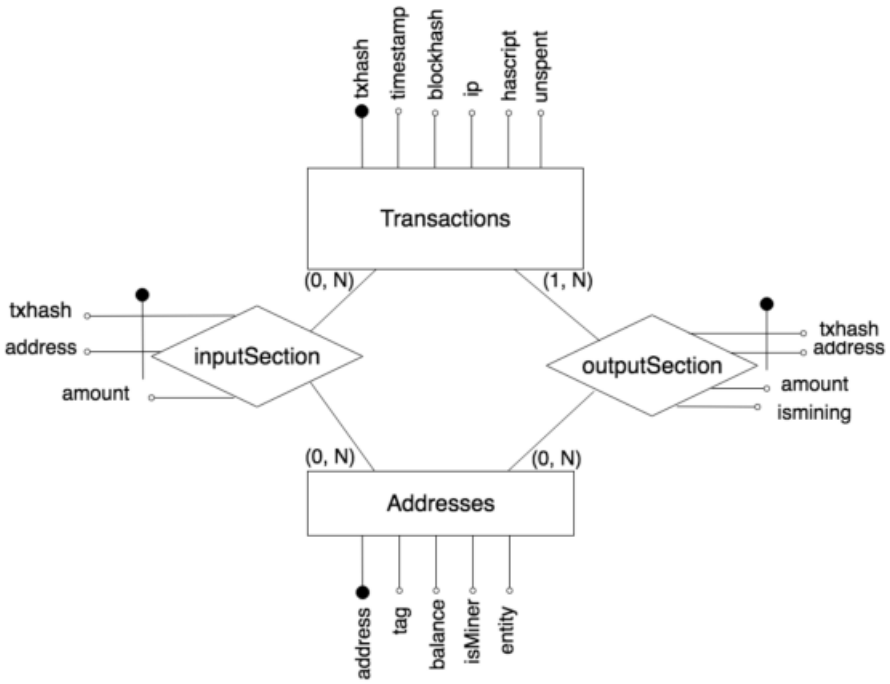
Algorytm aktualizacji (Updating Algorithm)

Algorithm 4: updateEntities(exAddr)

```
Input: A list exAddr of addresses already explored
j ← 0;
while j < length of exAddr do
    address ← exAddr[j];
    if (address is part of another entity) then
        select all the addresses of the different entity ;
        join the addresses with exAddr;
    end
end
```

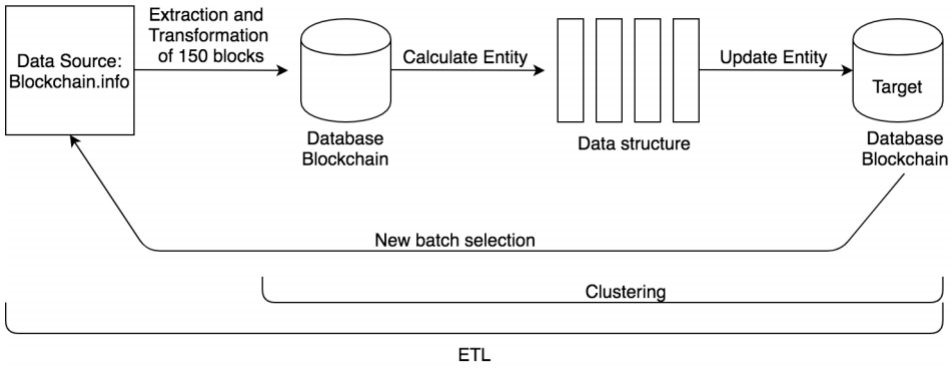
Rysunek 1.4: Pseudokod algorytmu aktualizacji [1]

Ładowanie (Load):
Ładowanie przetwarzanych danych będzie polegało na zamieszczeniu danych w odpowiednio zaprojektowanej relacyjnej bazie danych.
Architektura bazy danych:



Rysunek 1.5: E-R diagram relacyjnej bazy danych [1]

Poniżej jest przedstawiony schemat przebiegu procesu ETL:



Rysunek 1.6: Przebieg procesu ETL [1]

1.3 Wykorzystywane narzędzia

- Język programowania: Python
- Baza danych: PostgreSQL
- HTTP biblioteka: requests
- Operacje na haszach: hashlib
- Logowanie: logging
- Parser parametrów: argparse

Literatura

Artykuł

1. Applying the ETL Process to Blockchain Data. Prospect and Findings
Roberta Galici, Laura Ordile, Michele Marchesi, Andrea Pinna and Roberto Tonelli