information

MDPI

Article

Applying the ETL Process to Blockchain Data. Prospect and Findings

Roberta Galici [1], Laura Ordile [1], Michele Marchesi [1], Andrea Pinna [2,*] and Roberto Tonelli [1]

[1] Department of Mathematics and Computer Science, University of Cagliari, Via Ospedale 72, 09124 Cagliari, Italy; r.galici1@studenti.unica.it (R.G.); l.ordile@studenti.unica.it (L.O.); marchesi@unica.it (M.M.); roberto.tonelli@dsf.unica.it (R.T.)

[2] Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, Piazza D'Armi, 09100 Cagliari, Italy

* Correspondence: a.pinna@diee.unica.it

Received: 7 March 2020; Accepted: 7 April 2020; Published: 10 April 2020

Abstract: We present a novel strategy, based on the Extract, Transform and Load (ETL) process, to collect data from a blockchain, elaborate and make it available for further analysis. The study aims to satisfy the need for increasingly efficient data extraction strategies and effective representation methods for blockchain data. For this reason, we conceived a system to make scalable the process of blockchain data extraction and clustering, and to provide a SQL database which preserves the distinction between transaction and addresses. The proposed system satisfies the need to cluster addresses in entities, and the need to store the extracted data in a conventional database, making possible the data analysis by querying the database. In general, ETL processes allow the automation of the operation of data selection, data collection and data conditioning from a data warehouse, and produce output data in the best format for subsequent processing or for business. We focus on the Bitcoin blockchain transactions, which we organized in a relational database to distinguish between the input section and the output section of each transaction. We describe the implementation of address clustering algorithms specific for the Bitcoin blockchain and the process to collect and transform data and to load them in the database. To balance the input data rate with the elaboration time, we manage blockchain data according to the lambda architecture. To evaluate our process, we first analyzed the performances in terms of scalability, and then we checked its usability by analyzing loaded data. Finally, we present the results of a toy analysis, which provides some findings about blockchain data, focusing on a comparison between the statistics of the last year of transactions, and previous results of historical blockchain data found in the literature. The ETL process we realized to analyze blockchain data is proven to be able to perform a reliable and scalable data acquisition process, whose result makes stored data available for further analysis and business.

Keywords: ETL; Bitcoin; blockchain; lambda architecture; blockchain analytics

1. Introduction

This work presents the design and the implementation of a novel tool for blockchain analysis. A blockchain is a growing archive of blocks of digital records of money transactions, conceived by Satoshi Nakamoto [1] to create the Bitcoin payment system.

There are several reasons why it is interesting to analyze blockchain data. For example, we can see blockchain data as a public ledger of money transfers and theoretically this allows performance of a financial analysis and to forecast the behavior of investors in a market of about 150 billion dollars on data publicly available. In addition, by analyzing blockchain data it is possible to infer

on users' personal identities, by associating a tag or external information to pseudo-anonymous Bitcoin addresses.

Specifically conceived mixing algorithms of Bitcoin transactions [2,3], which allow a better privacy preservation, can be evaluated by analyzing blockchain data. Furthermore, blockchain data are available for forensic investigation [4,5]. Discovering the origin of scams, theft or abuse in the Bitcoin system allows maintenance of the blockchain's high reputation and trust. One of the most important elements in the blockchain analysis is the addresses clustering, which represents the starting point for several analyses.

Given the blockchain size rapid growth, it is necessary to conceive and implement increasingly efficient and scalable tools for blockchain data extraction and analysis. For this reason, we conceived a novel blockchain data extraction process with the objective to create a scalable and convenient system for data extraction, for clustering of addresses, and for making data usable for any type of analysis. Starting form recent research results, and given the need to design increasingly effective data extraction strategies and representation methods, the goal of this work is to realize a scalable process which stores the extracted data in a conventional database, making possible any typology of data analysis by querying it.

To achieve our goals, we conceived a system to extract blockchain data, transform and load it into a relational database, to make data available for further analysis. Starting from the requirement of scalability, we designed and implemented our system as an ETL (Extract, Transform, Load) process. The ETL process extracts a batch of 150 blocks at a time (about one day of transactions) from the blockchain. The transformation phase transforms blockchain data in the target data structure (four tables of the SQL database) and executes the algorithm for the addresses clustering. Finally, the load phase stores the transformed data in the target database. To achieve scalability we divided the transformation phase in two components. The fast access component transforms blockchain transaction data in four relationships (the addresses, the transactions, the input sections, and the output sections). The slow access component elaborates the batch entries to compute the address clustering. In this way the computational effort remains limited to the elaboration of the single batch, even if the database size increases.We evaluated the usability of the output data by replicating some of the statistical analysis performed in previous works and we obtained the same results by means of SQL queries, instead of processing ad hoc data structures.

The paper is organized as follows. Section 2 discusses the related works which deal with blockchain data analysis, privacy issues, and on deanonymization strategies and provides an overview of the technology used in our work i.e., the ETL process and the Lambda Architecture. Section 3 describes the architecture and the implementation of the system we conceived to extract and analyze blockchain data and, in particular, the role of the ETL process and of the Lambda Architecture in our system. In this section, we also focus our attention on the developed algorithms. Section 4 presents the results obtained by running our process and by querying the database. Section 5 discusses the implications of the results of our process and compares the data obtained by querying our database with the data obtained in previous works. Finally, we sum up results and discuss findings and implications in Section 6.

2. Related Works

A blockchain is a growing archive of blocks of digital records of money transactions, characterized to be unchangeable, publicly available, distributed and decentralized in a peer-to-peer network. The blockchain technology is the solution conceived by Satoshi Nakamoto to create the Bitcoin system and to solve the problem of double spending of digital money [1]. Bitcoin was conceived in 2008 but the research community started evaluating how and why to analyze blockchain data only a few years later, when the Bitcoin become popular. Presently, hundreds of public blockchains exist, one for each minable cryptocurrency. According to available data (coinmarketcap.com) the highest daily volume and highest capitalization still belong to the blockchain of Bitcoin. Almost unknown in the early years

of its existence, Bitcoin popularity surged in the end of 2013. Since then, it has never stopped growing, passing from about 5 GB of stored data in the beginning of 2013 to over 250 GB in the beginning of 2020 (source: blockchain.com). Contextually, the research community started to analyze blockchain data.

The first relevant scientific works regarding the analysis of blockchain data were published in 2013. The work of Ron and Shamir [6] is one of the first which organize a specific data structure to record transaction data and which allows a graph-based analysis. They used the concept of entity, which represents a cluster of Bitcoin addresses we can lead back to the same owner. The same year, Meiklejohn et al. [7] investigates on the traceability of payments in Bitcoin by analyzing the transaction data stored in the blockchain. They also evaluate the possibility to tracking down theft or scams, such as the Mt. Gox case. Reid and Harrigan [8] studied how it is possible to map coins movement by tracing user's addresses and by gathering information from other sources. They also showed that the network of address interactions by means of transactions follows the properties of complex networks. The transaction graph was also the focus of Ober et al. [9] which they studied to evaluate the statistical relation between the number of entities and the dimension (i.e., the number of addresses) of the entities.

More recently, researchers started a systematic investigation on Bitcoin user interrelations [10] and habits, i.e., the address usage and their strategies to improve the anonymization [2,3], vulnerabilities [11–13], and so on, and to allow forensic investigations, to, for instance, identify the origin of scams and illegal activities, such as the case of the blackmail WannaCry, with which criminals required Bitcoin payments to unlock the victim machines [4,5,14].

All these studies are made possible by novel strategies to perform the blockchain representation and the clustering of Bitcoin addresses in a more efficient way [15–18]. In particular, Pinna et al. [19] used a bipartite graph (represented as Petri net), to describe as nodes both entities and transactions and to allow performing investigations and statistics, and Bartoletti et al. [20] proposed a general framework to deeply analyze blockchain data properly stored in a database, by using the database query language. The use of SQL databases to represent blockchain data is also proposed by Yue et al. [21] and by the project Bitcoin Database Generator (https://github.com/ladimolnar/BitcoinDatabaseGenerator), while the project Bitcluster (https://github.com/mathieulavoie/Bitcluster/wiki/Database-structure) uses the noSQL database MongoDB.

We conceived our system aware both of the research results obtained to date, and of the need to continue to find increasingly effective data extraction strategies and representation methods. Aiming to realize a scalable process, we focused on the results of the three last mentioned research papers, and, in particular, on the need to preserve the distinction between transaction and addresses, the need to cluster addresses in entities, and the need to store the extracted data in a conventional database, making possible the data analysis by querying the database.

2.1. Background

Our system for blockchain data extraction and analysis is based on two technological solutions that allow efficient and scalable data processing. In particular, our system uses the process Extraction Transform and Load (ETL) and the Lambda architecture. For the sake of clarity, we retain useful to describe the basic concepts of these two technologies.

2.1.1. ETL Process

An ETL process is an ordered sequence of operations, namely Extract, Transform and Load, which aims at the systematic processing of source data in order to make them available in a format more convenient for the intended use.

The implementation of an ETL process requires a design phase, which according to Trujillo et al. [22], focuses on the following six tasks. (1) Selection of data sources and extraction: generally, the sources from which extract the data can be several and heterogeneous among them. (2) Join data from sources: definition of how to merge data from different sources. (3) Transform:

definition of how transform data. In this task are included: filtering data, calculating derived values, transforming data formats, etc. (4) Select the target: this task concerns the definition of the target (or targets) to which data will be loaded. (5) Attributes mapping: definition of the correspondence between the attributes of the source data with the attributes of the target. (6) Data loading: In this task the modality of how the target is populated with the transformed data is defined.

As mentioned, we defined our ETL process by following the tasks discussed above. Modeling the workflow of our system as an ETL process saved us design time and made the system easier to maintain. In particular, as will be described in the followings, we identified our data source as one of the available blockchain online resources, then we transformed the data by cleaning it from the all elements we do not need to represents. So, we selected and defined the target as a SQL database, and by specific algorithms we proceeded to load the data.

2.1.2. Lambda Architecture

Presented as a software design pattern, the lambda architecture unifies online and batch processing within a single framework. The pattern is suited to applications where there are time delays between data collection and their availability.

The pattern also allows batch processing for already stored data, useful for data elaborations which require a long computational time (such as behavioral patterns, etc.) [23,24].

There are several technologies for analyzing Big Data streams [25], but these do not allow the calculation of arbitrary functions on an arbitrary dataset in real time. To fix this problem it would be useful to use different techniques and tools to build a complete Big Data system. This problem could be divided into three layers of the Lambda Architecture, in particular, they are the batch layer, the serving layer and the speed layer. These three layers, which are the main components of the lambda architecture, respond to queries and manage to interact with the new input data.

We decided to use the Lambda Architecture according to the motivations given by Marz et al. [24], they summarize as following.

- *The need for a robust system that is fault-tolerant, both against hardware failures and human mistakes.*
- *To serve a wide range of workloads and use cases, in which low latency reads and updates are required. Related to this point, the system should support ad hoc queries.*
- *The system should be linearly scalable, and it should scale out rather than up, meaning that throwing more machines at the problem will do the job.*
- *The system should be extensible so that features can be added easily, and it should be easily debuggable and require minimal maintenance.*

3. Method

In this section, we describe the design and implementation of our system in terms of ETL process.

To design our system, we first defined two main requirement which transformed data should satisfy. The first requirement concerns the desired output of the ETL process: a SQL database containing blockchain data in a form which allows the same typologies of analysis performed in the literature, but by using SQL queries. The second requirement concerns the content of the database: it should contain both address, transactions and entities data. In other terms, the database should include the results of the address clustering.

To satisfy the two requirements we conceived the system represented in Figure 1. The overall process is composed by two elements: the overall ETL process, and the inner ETL process (Clustering) which is the slow process we conceived to satisfy the second requirement. In particular:

ETL: concerns the extraction, transformation and load of batches of 150 blocks at time, from the data source. The batch dimension is such to contain approximately one day of Bitcoin transactions. Fixed size batch extraction allows satisfaction of the requirement of scalability, because it fixes the amount of data and limits the processing times. In this process, the data arrives quickly to the data warehouse, thanks to the use of the processing speed of the lambda architecture. The extraction phase

is followed by the cleaning and transformation phase. Then data are mapped and finally, for the loading phase, the final data is saved in the target database.

Clustering: is the internal ETL process that deals with the identification of entities. It is used by the batch processing phase of the lambda architecture, which has a slow access because it queries the database and therefore data extraction is much slower, due to the most complex computational process. For the extraction phase, it fetches the data related to the last batch of 150 blocks from the database. The transformation phase concerns the clustering of addresses in entities, as will be discussed in the following. Finally, for the loading phase, the process updates the database by writing for each address the corresponding entity.

To realize our system, we implemented the ETL process and the Clustering algorithm by using python 2.7 as programming language and postgreSQL as DBMS. The source codes are online available at https://github.com/ETL-Blockchain/ETL-process.

For both the two elements of the process we will describe design and implementation.

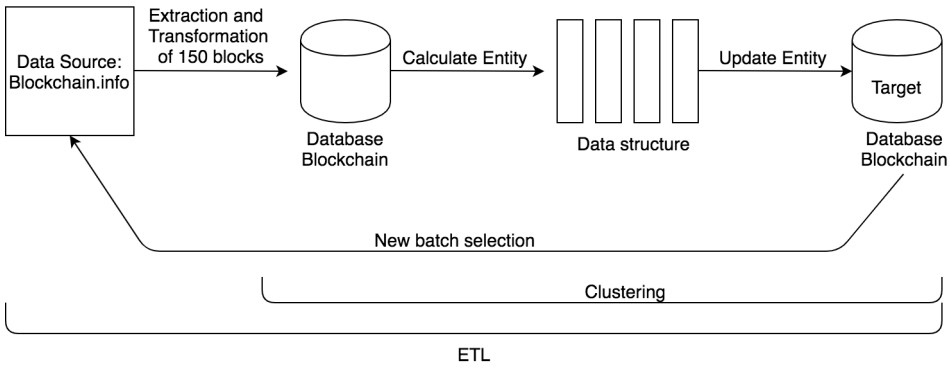

Figure 1. ETL workflow representation for blockchain data processing.

3.1. ETL Process Design

The first task of the design of the ETL process concerns the selection of data sources and the data extraction. The data source of our process was selected between one of the free online available blockchain explorers. Each blockchain explorer provides an API which returns blockchain data as serialized data (typically in Json format), useful for the remote analysis. For the choice of the blockchain explorer we have evaluated two parameters: the maximum number of requests per second and the completeness of available data. We analyzed four blockchain explorer:

- blockchair.com: 0.5 requests per second;
- blockcyper.com: 3 requests per second;
- blockchain.info: 5 requests per second;
- chain.so: 5 requests per second.

As can be seen from the list above, the fastest sources are those of blockchain.info and chain.so. Between the two, we chose to use blockchain.info because provides more complete data and because chain.so is still in development.

The next step regards the definition of the Data Warehouse, target of the ETL process. The target database should contain all the information the analyzer could need for the several typologies of investigations we previously mentioned. We conceived our database to be adaptable and upgradable if the analyzer would require further blockchain data. Aiming at a more efficient data representation, we decomposed blockchain transaction data in a total of four tables. A typical Bitcoin transaction is composed of two lists: a list of sender addresses and a list of receiver addresses. These lists are respectively called input section and output section of the transaction. To keep the completeness of transaction data, we record the input section and the output section of each transaction in two specific relations between addresses and transactions. Figure 2 shows the Entity Relationship diagram of the target database. The four tables contain the following data.

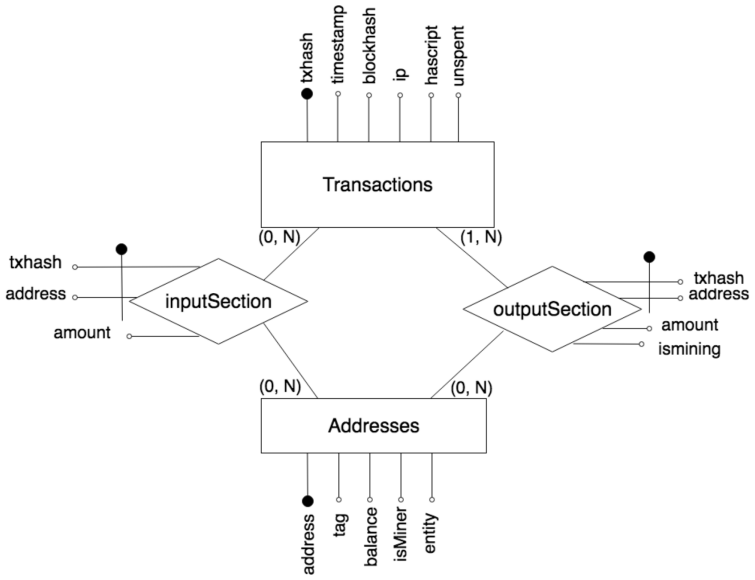

Figure 2. E-R diagram of the relational database target of the transformed data.

- Transactions (txhash (primary key), timestamp, block, IP, hasScript, unspent). Each tuple of this table is a blockchain transaction. Attributes of this table include: the transaction hash, the transaction date, the hash of the block which contains the transaction, the IP address of the miner, and two Boolean to represents if the transaction contains a script in its body, and if the transaction was unspent in the moment of the extraction.
- Addresses (address (primary key), tag, balance, IsMiner, entity). Each tuple of this table represents a blockchain address. Attributes include: the address, a tag (a short description, if available), the balance in Satoshi (1 Satoshi = 10^{-8} Bitcoin. It is the smallest amount of cryptocurrency to transfer with a transaction.) (which is computed as the sum of the total value of the unspent transactions received by the address), a Boolean to specify if it is was a miner at least one time, and the ID of the entity we computed in the clustering process (initially empty).
- InputSection (address (external key), txhash (external key), amount). Each tuple of this table represents one element of the input section of a given transaction. For instance, if one transaction contains three input, three entries will be recorded in this table. Attributes include: the address (which references to the table addresses), the transaction hash (which references to the table transactions), and the amount of Bitcoin in Satoshi.
- OutputSection (address (external key), txhash (external key), amount, isMining). Each tuple of this table represents one element of the output section of each transaction. Attributes include: the address (which references to the table addresses), the hash of the transaction (which references to the table transactions), the amount of money in Satoshi, and a Boolean to represent if a given address is the receiver of the mining prize obtained with a given transaction.

The use of a relational database constitutes an advantage because we must model very structured data such as the blockchain data. In such context, ductility of not relational databases does not appear to be a fundamental requirement. As mentioned, nothing prevents us to add fields to tables or add new tables, if and when needed.

We included the possibility to add a tag to a given address. It could be filled by adding information we can retrieve from blockchain.info and from several external sources. Tags and any external information, joined with entity clustering, we will discuss below, could be used to better recognize users' habits and to infer users' identity.

3.2. Clustering

As previously described, the Clustering process is the slow access component of the process, and it can be represented as an internal ETL process which takes as inputs the batch of 150 blocks and the blockchain data already stored in the SQL database (the sources of the extraction), computes the entities (transformation), and updates the database (load), final target of the process. This process implements a clustering algorithm based on the bipartite graph search [19] in which transactions and addresses represents the two families of nodes. A cluster is the set of Bitcoin addresses which belongs to the same owner (or entity).

The heuristic behind our algorithm, defined as the Heuristic "Multi-Input" [26] which establishes that the input addresses of a transaction always come from the same wallet, then the author of the transaction has the private keys of all the input addresses.

The Clustering Algorithm, reported in Algorithm 1 uses five lists to organize computational data and processes data contained in the table inputSection of the database. Lists contains: the transactions to be explored (*tr*), the transactions explored (*exTr*), the addresses to be explored (*addr*), the addresses explored (*exAddr*), and the output table of clusters (*ec*), in which row represents a cluster and its list of addresses. All the lists are empty. This algorithm will be executed for each new batch. Therefore, if no entity has been identified yet and we are at the beginning of the clustering, the first entity will be identified with and ID equal to 1. Otherwise the next ID will be the next value of the maximum id of the entities identified until that moment. Given a batch of 150 blocks, the algorithm takes in input the related selection of tuples extracted from the table *inputSection* of the database. The algorithm starts by taking the first tuple of the selection and starts the exploration of the transaction in which the first unexplored address is involved. This process is described in Algorithm 2 and explained in the following. It returns new unexplored transactions which are added to *tr*. Then, to find the other addresses belonging to the same owner, the algorithm explores the first unexplored transaction in *tr* and adds its input addresses to *addr*. This process is described in Algorithm 3 and explained in the following.

Figure 3 shows the working principles of the Clustering algorithm. In particular, given the elements in the "input section" and the initial state, the algorithm execute the Transaction search (TS) for the first unexplored address in *addr* and then, executes the Address search (AS) for the first unexplored transaction in *tr*. Finally, the algorithm returns *er*, the cluster of addresses. The example in Figure 3 is conceived to represent the creation of only one cluster containing all the addresses present in the input section.

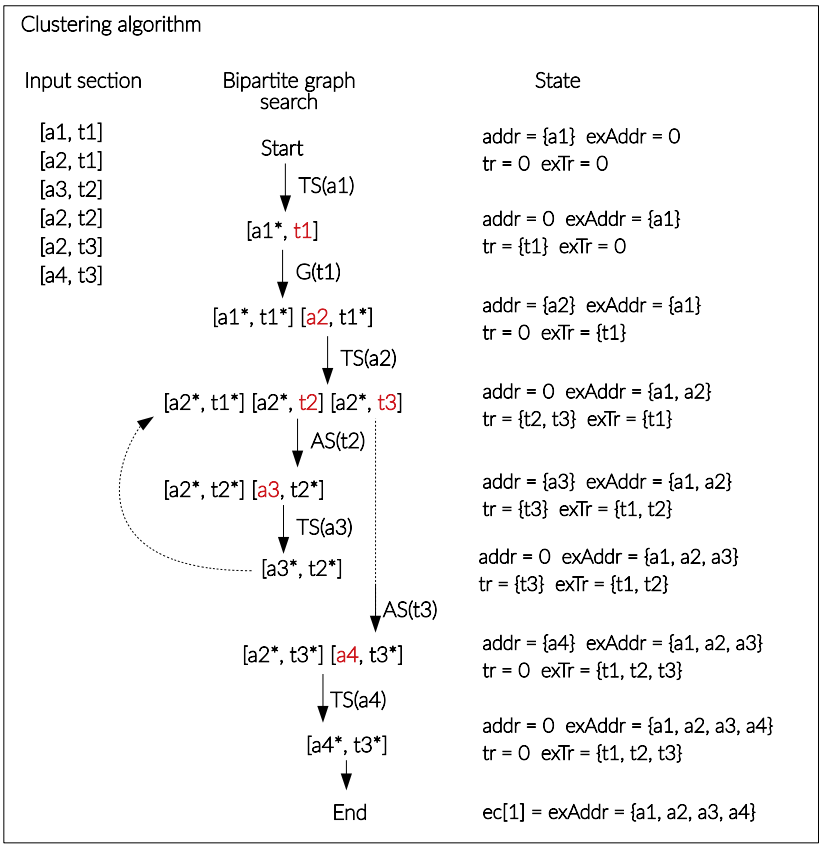

Figure 3. Graphical example of working principles of the Clustering algorithm. In this figure, the input section represents the content of the resulting table inputSection obtained after the transformation of a batch of 150 blockchain blocks. Letters *a* and *t* represent the values of the fields *address* and *transaction hash* contained in this table. The symbol TS is the transaction search algorithm (Algorithm 2), and the symbol AS is the address search algorithm (Algorithm 3).

Once the batch of 150 blocks are processed and entities have been created, the algorithm updates the field *entity* of the table *addresses*. It queries the database to join the new clusters with existent entities, and contextually update the entities IDs if necessary. This process is described in Algorithm 4 and will be explained in the following.

The clustering algorithm includes the following sub-algorithms.

- **Transaction search algorithm (Algorithm 2)**: it takes in input an unexplored address and searches all the transactions where the unexplored address is present, and once the transactions have been found, the address is removed from the list of addresses to be explored.
- **Address search algorithm (Algorithm 3)**: takes in input an unexplored transaction and searches all the addresses present in the input section of the transaction under consideration, excluding those already explored. At the end of the process it marks the transaction as explored and adds the result of the search to the queue of addresses to be explored.
- **Updating Algorithm (Algorithm 4)**: If the same address has been clustered into multiple entities, these clusters are joined into a single entity, identified by a new ID equal to the existing maximum ID incremented by one. Contextually, the algorithm updates the table addresses of the database which, for each address, writes the new entity ID.

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;
```

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;
```

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*;

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

4. Results

In this section, we report the results of the experimental usage of our ETL process for blockchain data we performed to evaluate our system.

The first part of the results concerns the process performance. For each batch, the first part of the transformation phase (namely the mapping of blocks raw data to the database tables) requires a time less than the downloading time of blocks data, whose size is in the hundreds of MB (about 150 MB per batch). As expected, the clustering algorithm slows the whole process down, but, given the limited number of data contained in each batch, the determination and updating of the entities is completed in hours and the time grows less than the total size of extracted data. Figure 4 shows the cumulative number of examined address-transaction rows of the table inputSection versus the processing time of the first ten batches (1500 blocks; 3,244,034 addresses-transaction inputs in 68,639 s). It is possible to note that the total number of processed addresses-transaction linearly increase with the computational time. The processing time of each batch essentially only depends on the batch dimension. On average, the process compute 48.6 rows per second and each batch is processed in 6864 s. Figure 5 shows the processing time of each one of the first ten batches, compared with the number of rows in the input sections. In our experimental setup, we used a PC equipped with a six-core CPU at 2.20 GHz and 16 GB of RAM.

The second part of the results concern the evaluation of the effectiveness of the database to analyze blockchain data. We collected a total of 142 batches of 150 blocks, equal to a total of 21,300 blocks. The final disk size of the database is around 54 GB. At the end of the process, we were able to study blockchain data, for example to obtain results and usage statistics, simply by querying the database with SQL queries.

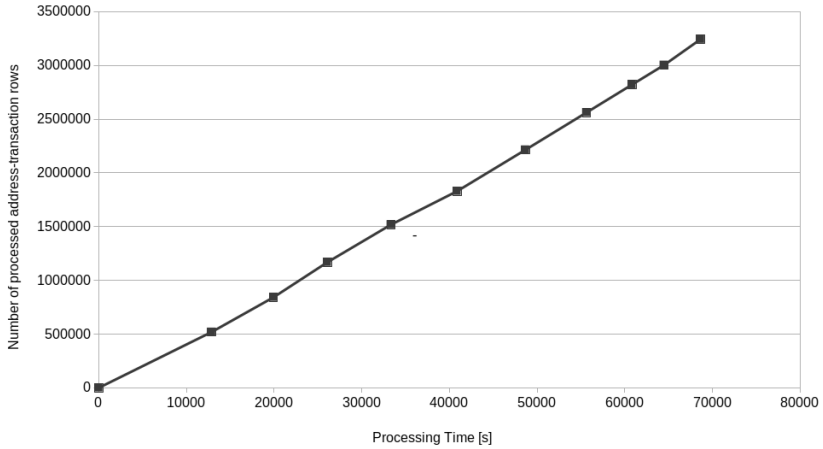

Figure 4. Number of examined address-transaction rows of the table inputSection versus the ETL processing time.

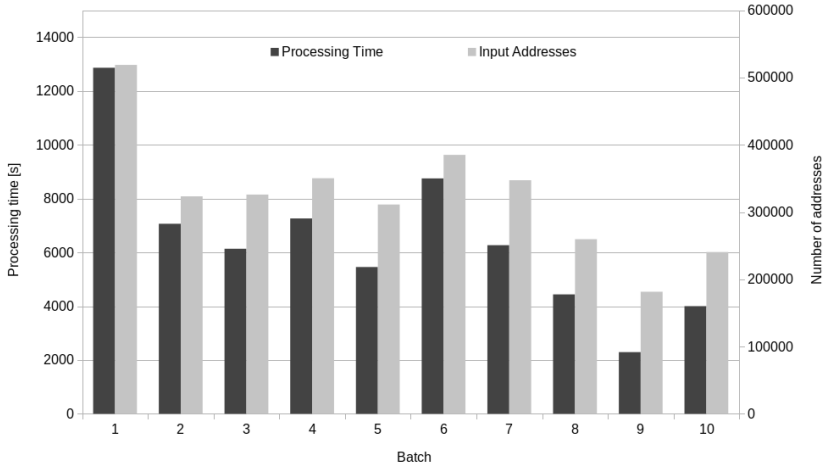

Figure 5. ETL processing time for the first 10 batches (left axis) and the corresponding number of addresses-transaction rows in the table inputSection (right axis)

For example, we wrote the following query to obtain the average value of how many times each address is used to send transactions.

```
SELECT AVG(n_txs)
FROM( SELECT COUNT(txhash) AS n_txs
FROM inputsection GROUP BY address )
```

We wrote the following query to obtain the maximum size of addresses clusters (namely the maximum number of addresses contained in an entity).

```
SELECT MAX(n_address)
FROM( SELECT count(address) AS n_address
FROM addresses GROUP BY entity )
```

Table 1 reports statistics and findings we obtained by querying our database. In particular, we queried the database to obtain: the number of transactions stored in the database (namely the number of rows of the table *transactions*), the number of unique addresses (namely the number of rows of the table *addresses*), the average number of transactions which any address send (obtained with the first query reported above), the number of records of the table inputSection (which is equal to the number

of couples address-transaction analyzed by the clustering algorithm), the average length of the input section per transaction, and the average value of Bitcoin transactions. For what concerns the entities, we queried the database to obtain: the total number of entities (which contains at least two addresses), the maximum (see the last query above) and the average number of addresses contained in a entity.

The first columns of the table contain the preliminary results obtained by analyzing the first batch (150 blocks) which allowed us to validate the process by a punctual verification of the resulting data. The second column contains the results of the full data set we created (21,300 blocks). The third column contains the results of the same statistics but related to the oldest 180,000 blocks of the blockchain, as obtained in [19] by analyzing an ad hoc data structure instead of a querying a SQL database. In the next section we will compare these old results with the most recent ones we obtained in this work. It must be noted that our process collects data going backwards in time. The highest block height we set is the block number 566,874 (mined the 13Th March of 2019).

Table 1. Blockchain statistics obtained by querying the database. This table reports the results of the queries applied to the single batch (150 blocks) and to a larger collection of 21,300 blocks (142 batches), starting from the maximum height of 566,874. As a term of comparison, the table reports the results of the same queries for the first 180,000 blocks of the blockchain. For this column, the reported values are computed on the base of findings presented in [19]

	150 blocks	**21,300 blocks**	**180,000 blocks** [19]
Disk space	449 MB	54 GB	-
Number of Transactions	163,442	30.99×10^6	3.14×10^6
Number of Addresses	547,553	56.50×10^6	3.73×10^6
Average number of transactions sent by an address	1.580	1.508	1.2266
Total number of entries in the table inputSection	584,409	48.7×10^6	4.57×10^6
Average length of the input section per transaction	3.5789	2.695	1.4564
Average amount in Satoshi in the input sections	169.7×10^6	46.32×10^6	-
Number of Entities	128,207	6.54 x 10^6	2.46×10^6
Maximum number of addresses per entity	177,768	4.072×10^6	156,725
Average number of addresses per entity	4.270	4.1526	1.5158

5. Implications

The results obtained in the previous section allow us to state that the ETL process we designed and implemented achieves both objectives of this research work. In particular: performance results show us that the ETL system scales as expected, and this allows you to estimate the time required to extract and cluster a given number of batches or for a given number of blocks; queries results reported

in Table 1 shows the effectiveness of our process to provide blockchain data from the SQL database created by means of our ETL process, by means of SQL queries.

By analyzing the results of the queries reported in Figure 1 we can figures out the current state of the blockchain (updatet to March 2019), in terms of addresses interconnection and of users' habits.

Each block introduces a much higher number of new addresses and new transactions, compared to what happened during the first 180,000 blocks. This is due on the higher popularity of Bitcoin and the usual reaching of the limit of 1 MB per block imposed by the current protocol which limits the number of transactions. More interesting is the comparison of the ratio between the number of addresses and the number of transactions because reveals changes in users' habits. In the most recent 21,300 blocks, this ratio is equal to 1.825. In the first 180,000 blocks this ratio was equal to 1.187. Then, the number of addresses increases faster than the number of transactions. The difference in addresses usage are reflected in the subsequent value in the table, namely average number of transactions sent by each address (considering addresses involved as input in at least one transaction). Currently, each address is used to send, on average about 1.5 transactions, while in the past, each address performed little more than one transaction. This means that more addresses are reused in two or more transaction than in the past.

The average number of addresses involved as input in each transaction is currently equal to 2.695. This value is increased with respect the same value computed for the first 180,000 blocks and equal to 1.456. This could be the effect of the mixing algorithms with which Bitcoin users protect their privacy.

We then analyzed the entities, result of the clustering algorithm. The total number of entities we found is equal to 6.45 million. We can observe that this number is only 3 times higher of the value found for the first 180,000 blocks, but these new entities contain more addresses. Each entity obtained with our process contains more than four addresses on average, while in the first 180,000 blocks there were about 1.5 addresses per entity. This difference can be explained considering two aspects. The first aspect concerns the fact that currently more Bitcoin users' received Bitcoin in more than one of their addresses. So, the total amount of Bitcoin owned by a user is distributed in multiple addresses, and the user has, in turn, to input more addresses to pay a certain amount of Bitcoin. For the second aspect, we can take into account the massive use of transaction mixers (as described in Section 2) which create fake entities that are able to obfuscate the effective cryptocurrency flow, from the sender to the receiver. This can be the reason we individuated a cluster of over 4 million addresses. This aspect should be taken into account in any deanonymization processes.

6. Conclusions

This work describes the motivation, the design and the implementation of an ETL process applied to the blockchain of Bitcoin. The process we realized satisfies the requirement of scalability and the requirement of providing in output the structured data available for further analysis. To achieve our goals, we organized the process in the three phases of the ETL process. In particular, the extraction phase of the process fetches batch of 150 blocks raw data each time, from an online source; the transformation phase organizes raw data to be stored into the target database and computes the clusters of address called entities; the load phase store data onto a SQL database which also distinguishes between the input section and the output section of each transaction. We organized our ETL process by using the Lambda Architecture which distinguishes between the fast access process (our extraction phase, the mapping and the loading of transformed data in the database), and the slow access process (the clustering algorithm and the updating of the database). After implemented our process, we first assessed the fulfillment of the scalability objective, by monitoring the performance of an analysis over 1500 blocks (10 batches). We thus found that the ETL process performance is not affected by the growing size of the database but maintains batch processing time essentially dependent on the contents of the batch itself.

Then we assessed the usability of the designed database by executing the processing of 142 batches (starting from the block 566,874 and going back for 21,300 blocks) and querying the resulting database

by means of SQL instructions. In this occasion we found several changes of the Bitcoin users' habits, in comparison with the statistics related to the first 180,000 blockchain blocks. All further analyses can be performed through SQL queries. The database we designed allows tracking of the activities of the entities, evaluate the monetary flows, and, through cross-referenced data, trace the identities of the Bitcoin users. This work demonstrates the effectiveness of our approach. We will proceed with the expansion of the database until reaching the total number of blocks given that the proposed ETL approach scales smoothly with data size. Further studies will allow regulation of the process, and in particular the batch size, to maximize performance.

Author Contributions: Conceptualization, R.G., L.O., M.M., A.P and R.T; methodology, A.P. and R.T.; software, L.O.; validation, A.P. and R.T.; formal analysis, A.P.; investigation, R.G.; data curation, A.P. and L.O.; writing–original draft preparation, R.G. and L.O; writing–review and editing, A.P. and R.T.; supervision, M.M. and R.T.; project administration, M.M. and R.T. All authors have read and agree to the published version of the manuscript.

Funding: This research was partially funded by Regione Autonoma della Sardegna, under project "CAFCha"—Programmazione unitaria 2014–2020 POR FESR Sardegna 2014–2020, and under project "Crypto-Trading"—Programmazione unitaria 2014–2020 POR FESR Sardegna 2014–2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. A Peer-to-Peer Electronic Cash System. Bitcoin.org. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 7 March 2020).
2. Maxwell, G. CoinJoin: Bitcoin Privacy for the Real World. bitcointalk.org. 2013. Available online: https://bitcointalk.org/?topic=279249 (accessed on 7 March 2020).
3. Ruffing, T.; Moreno-Sanchez, P. Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin. In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017; pp. 133–154.
4. Bistarelli, S.; Mercanti, I.; Santini, F. A suite of tools for the forensic analysis of bitcoin transactions: Preliminary report. In Proceedings of the European Conference on Parallel Processing, Turin, Italy, 27–31 August 2018; pp. 329–341.
5. Wu, Y.; Luo, A.; Xu, D. Forensic Analysis of Bitcoin Transactions. In Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), Shenzhen, China, 1–3 July 2019; pp. 167–169.
6. Ron, D.; Shamir, A. Quantitative analysis of the full bitcoin transaction graph. In Proceedings of the International Conference on Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; pp. 6–24.
7. Meiklejohn, S.; Pomarole, M.; Jordan, G.; Levchenko, K.; McCoy, D.; Voelker, G.M.; Savage, S. A fistful of bitcoins: characterizing payments among men with no names. In Proceedings of the 2013 Conference on Internet Measurement Conference, Barcelona, Spain, 23–25 October 2013; pp. 127–140.
8. Reid, F.; Harrigan, M. An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*; Altshuler, Y., Elovici, Y., Cremers, A.B., Aharony, N., Pentland, A., Eds.; Springer: New York, NY, USA, 2013; pp. 197–223. doi:10.1007/978-1-4614-4139-7_10. [CrossRef]
9. Ober, M.; Katzenbeisser, S.; Hamacher, K. Structure and anonymity of the bitcoin transaction graph. *Future Internet* **2013**, *5*, 237–250. [CrossRef]
10. McGinn, D.; Birch, D.; Akroyd, D.; Molina-Solana, M.; Guo, Y.; Knottenbelt, W.J. Visualizing dynamic bitcoin transaction patterns. *Big Data* **2016**, *4*, 109–119. [CrossRef] [PubMed]
11. Harrigan, M.; Fretter, C. The unreasonable effectiveness of address clustering. In Proceedings of the 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, 18–21 July 2016; pp. 368–373.
12. McGinn, D.; McIlwraith, D.; Guo, Y. Towards open data blockchain analytics: a Bitcoin perspective. *R. Soc. Open Sci.* **2018**, *5*, 180298. [CrossRef] [PubMed]

13. Zheng, B.; Zhu, L.; Shen, M.; Du, X.; Guizani, M. Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering. *Sci. China Inf. Sci.* **2020**, *63*, 1–15. [CrossRef]
14. Zheng, B.; Zhu, L.; Shen, M.; Du, X.; Yang, J.; Gao, F.; Li, Y.; Zhang, C.; Liu, S.; Yin, S. Malicious bitcoin transaction tracing using incidence relation clustering. In Proceedings of the International Conference on Mobile Networks and Management, Melbourne, Australia, 13–15 December 2017; pp. 313–323.
15. Ermilov, D.; Panov, M.; Yanovich, Y. Automatic bitcoin address clustering. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 461–466.
16. Shin, M.G.; Baek, U.J.; Shim, K.S.; Park, J.T.; Yoon, S.H.; Kim, M.S. Block Analysis in Bitcoin System Using Clustering with Dimension Reduction. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 18–20 September 2019; pp. 1–4.
17. Neudecker, T.; Hartenstein, H. Could network information facilitate address clustering in Bitcoin? In Proceedings of the International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017; pp. 155–169.
18. Maesa, D.D.F.; Marino, A.; Ricci, L. Data-driven analysis of Bitcoin properties: exploiting the users graph. *Int. J. Data Sci. Anal.* **2018**, *6*, 63–80. [CrossRef]
19. Pinna, A.; Tonelli, R.; Orrú, M.; Marchesi, M. A Petri Nets model for blockchain analysis. *Comput. J.* **2018**, *61*, 1374–1388. [CrossRef]
20. Bartoletti, M.; Lande, S.; Pompianu, L.; Bracciali, A. A general framework for blockchain analytics. In Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, Las Vegas, NV, USA, 11–15 December 2017; pp. 1–6.
21. Yue, K.B.; Chandrasekar, K.; Gullapalli, H. Storing and Querying Blockchain using SQL Databases. *Inf. Syst. Educ. J.* **2019**, *17*, 24.
22. Trujillo, J.; Luján-Mora, S. A UML based approach for modeling ETL processes in data warehouses. In Proceedings of the International Conference on Conceptual Modeling, Chicago, IL, USA, 13–16 October 2003; pp. 307–320.
23. Kiran, M.; Murphy, P.; Monga, I.; Dugan, J.; Baveja, S.S. Lambda architecture for cost-effective batch and speed big data processing. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 2785–2792.
24. Marz, N.; Warren, J. *Big Data: Principles and Best Practices of Scalable Real-Time Data Systems*; Manning Publications Co.: New York, NY, USA, 2015.
25. Hasani, Z.; Kon-Popovska, M.; Velinov, G. Survey of technologies for real time big data streams analytic. In Proceedings of the 11th International Conference on Informatics and Information Technologies, Las Vegas, NV, USA, 7–9 April 2014; pp. 11–13.
26. Androulaki, E.; Karame, G.O.; Roeschlin, M.; Scherer, T.; Capkun, S. Evaluating user privacy in bitcoin. In Proceedings of the International Conference on Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; pp. 34–51.