UNIVERSITÀ DEGLI STUDI DI VERONA

Corso di Laurea in
Informatica

Elaborato finale

EtherChain2DB:
Un framework per la memorizzazione e l'analisi integrata della blockchain di Ethereum tramite un database relazionale.

Relatore
DR. SARA MIGLIORINI

Candidato
DE BONI SIMONE
Matricola VR421790

Anno Accademico 2019 – 2020

SOMMARIO

1. INTRODUZIONE

L’obiettivo di questo elaborato è la definizione e la creazione di una base di dati relazionale in grado di memorizzare le informazioni presenti e riguardanti la blockchain Ethereum, per poi poterle studiare attraverso delle interrogazioni scritte in linguaggio SQL.
Questa tesi comincia spiegando alcuni aspetti generali riguardante il concetto di blockchain, mining e altri aspetti riguardanti questo ambito. Dopodiché verrà esposta più nel dettaglio la blockchain di Ethereum e tutte le caratteristiche che la contraddistinguono dalle altre blockchain di criptovalute, ad esempio le dApps, il gas e gli smart contracts, trattati solo in breve poiché argomento di tesi del mio collega con cui ho svolto il framework.
In seguito verrà esposto il modello ER della base di dati creata e verranno mostrati anche tutti i campi delle tabelle presenti nel database. A seguito verranno mostrate tutte le librerie testate per poter raggiungere il nostro obiettivo e spiegato il funzionamento del programma con cui è stata creata la base di dati su PostgreSQL, ovvero un programma scritto in Python composto da vari file che utilizzano delle librerie in grado di estrarre i dati dalla blockchain e un’altra libreria che carica le informazioni sul database.
Infine verranno mostrate alcune statistiche e grafici ricavabili anche attraverso alcune query eseguite sulla nostra base di dati.

2. INTRODUZIONE ALLA BLOCKCHAIN

2.1 Definizione di Blockchain e di blocco

Una blockchain è una struttura dati utilizzata per la gestione e memorizzazione di transazioni distribuite e replicate tra più nodi di una rete, suddivisa in blocchi concatenati tra loro in ordine cronologico.
Il blocco è un registro che raccoglie delle transazioni , nella parte chiamata body e diversi campi di gestione del blocco stesso, raccolti nel cosiddetto header, tra cui il proprio timestamp e un puntatore hash al blocco precedente.
Facendo parte della famiglia delle Distributed Ledger, una delle sue caratteristiche principali è quella di essere decentralizzata e distribuita massivamente tra tutti i partecipanti, chiamati nodi. Quindi ogni utente può essere in grado di avere la propria copia della blockchain salvata in memoria. Questi utenti sono considerabili tutti allo stesso livello di credibilità, difatti non esiste una vera e propria copia ufficiale.[1]
Un'altra caratteristica della tecnologia blockchain è l'utilizzo della crittografia asimmetrica per permettere lo scambio delle criptovalute tra due utenti. Ogni utente possiede due chiavi: quella pubblica, che corrisponde all'indirizzo, visibile da tutti gli altri utenti e quella privata, conosciuta solo dall'utente che ne è proprietario. Questo sistema permette a tutti di poter inviare qualcosa a un utente, ma solo all'utente destinatario di poterlo leggere. Se un utente deve inviare dei token, questi verranno criptati usando la chiave pubblica del destinatario e saranno decriptabili solamente dal destinatario, unico possessore della chiave privata. [2]
La blockchain viene anche definita immutabile poiché il suo contenuto, una volta scritto, non è più modificabile, se non attraverso l'invalidazione di tutta la struttura.
Una delle altre caratteristiche delle blockchain che può creare problemi è la sua dimensione nel tempo. La crescita esagerata di una blockchain può portare ad un aumento del rischio di centralizzazione dei nodi, dato che le risorse richieste per lavorare su di essa crescerebbero in modo esponenziale e non tutti gli utenti potrebbero essere in grado di avere un dispositivo in grado di poter soddisfare quei requisiti.

[1] https://blog.osservatori.net/it_it/distributed-ledger-technology-significato
[2] https://www.criptoinvestire.com/come-funziona-la-crittografia-nelle-blockchain.html

2.2 Creazione dei blocchi e concetto di Mining

Per garantire la coerenza tra le varie copie, le modifiche al contenuto vengono regolate tramite algoritmi di consenso. Tali algoritmi permettono di raggiungere il consenso tra le varie versioni del registro, nonostante esse siano aggiornate in maniera indipendente dai partecipanti della rete. Ogni nodo della rete può verificare autonomamente se una versione del blocco che riceve da un altro utente sia valida. Se lo è, il nodo lo aggiunge alla propria versione della catena, procedendo alla creazione del blocco successivo. Potrebbe succedere che due o più nodi generino contemporaneamente più blocchi validi, creando una diramazione, in inglese fork. Le diramazioni spariranno nel momento in cui un altro blocco al livello successivo verrà scoperto, così facendo si manterrà valida solo la catena più lunga e le altre diramazioni verranno scartate.
La creazione di blocchi, il cui processo viene chiamato mining, è caratterizzato da un algoritmo chiamato Proof of Work, ovvero un insieme di operazioni costose dal punto di vista elaborativo, che vengono svolte dal richiedente di un servizio per dimostrare che sia stato impiegato del tempo per generare la richiesta. Nel caso delle blockchain spesso si tratta di diversi rompicapi matematici o operazioni crittografiche con hash. Ciò viene fatto per evitare che ci siano troppe richieste, obbligando quindi il cliente a perdere del tempo per effettuarne una.
Ogni utente genera un numero (chiamato nonce), da inserire nel blocco, e questo valore deve fare in modo che l'hash del blocco risulti minore di un valore prefissato, detto target. L'unica strada per poter effettivamente generare un hash che rispetti questa condizione è generare nonce casualmente finché non si ottiene il risultato sperato. A quel punto il blocco viene creato, trasmesso alla rete e inserito nella blockchain. Chiunque potrà verificare la veridicità dell'hash, perché dato il numero iniziale, il calcolo di un hash è molto semplice. Chi riesce a trovare un hash minore del target viene premiato dal sistema con una ricompensa in criptovalute.

3. LA BLOCKCHAIN ETHEREUM

3.1 Introduzione su Ethereum

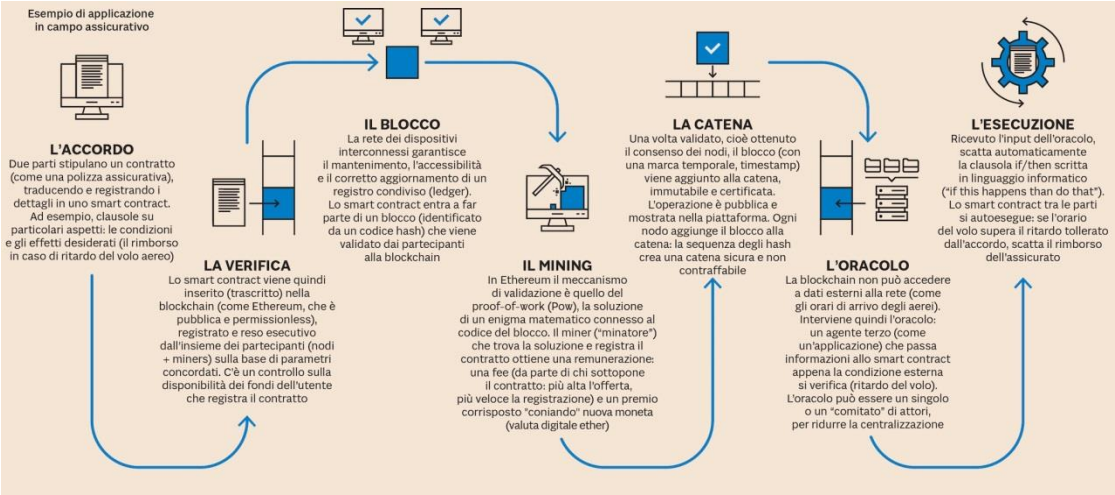
Ethereum è una delle più grandi e popolari blockchain basate su criptovalute. Creata tra il 2013 e il 2014 da Vitalik Buterin attraverso una raccolta fondi, è seconda solo a Bitcoin nella capitalizzazione azionaria tra le criptovalute. La dimensione della sola blockchain si aggira attualmente attorno ai 300GB e conta più di 10 milioni di blocchi.

Logo di Ethereum

Una delle caratteristiche che contraddistingue maggiormente questa Blockchain dalle altre è la possibilità di creare e gestire gli Smart Contract. Essi funzionano solo su Ethereum e vengono realizzati mediante dei linguaggi di programmazione specifici, i più noti sono Solidity e Viper. La principale caratteristica di questo tipo di contratti è che permettono l'avvenimento di determinate azioni, ad esempio transazioni da un account a un altro, solo se le condizioni e clausole su cui si basano trovano un riscontro nella realtà, ad esempio il risultato di una partita o l'eventualità di un ritardo di un volo.

Dato che la blockchain non può accedere ai dati esterni, le informazioni per le clausole degli smart contract vengono fornite dagli oracoli, servizi di terze parti che offrono per l'appunto un collegamento col mondo fornendo informazioni rilevanti. Senza di essi gli Smart Contract sarebbero limitati ai dati all'interno della rete della blockchain. Alcuni oracoli hanno l'abilità di non solo trasmettere informazioni agli smart contract, ma anche di rimandarle alle fonti esterne.[3]

[3] https://academy.binance.com/it/articles/blockchain-oracles-explained



Schema di creazione e di inserimento nella blockchain di uno smart contract (immagine tratta da un articolo del Sole 24 Ore)

Come per tutte le criptovalute, anche Ethereum presenta sia vantaggi che svantaggi nell'utilizzo. I principali vantaggi sono:

1. Libertà di pagamento: è possibile ricevere ed inviare denaro in qualsiasi momento ed in qualsiasi luogo. È sufficiente avere a disposizione uno smartphone o un computer con una connessione ad internet. Non essendo controllata o gestita da nessun ente, non è soggetta a manipolazione o interruzioni volontarie. Inoltre si possono effettuare transazioni anche durante le feste o di notte e l'invio e la ricezione di denaro sono istantanee.
2. Meno rischi: tutte le transazioni Ethereum sono sicure, irreversibili e non contengono dati personali. I contratti creati all'interno della rete non possono essere modificati da nessuno. Questa caratteristica rassicura le aziende e gli utenti stessi.
3. Trasparenza: tutte le informazioni sulle transazioni sono disponibili sulla blokchain in tempo reale. Nessun privato o organizzazione può controllare o manipolare il protocollo Ethereum perché è sicuro dal punto di vista crittografico.
4. Conveniente: rispetto ad altri sistemi di pagamento (banche, Paypal, Bitcoin), Ethereum è più economico rispetto ad altre piattaforme con commissioni alte.

Gli svantaggi invece sono:

1. Utilizzo: nonostante sia un metodo rivoluzionario, Ethereum non è utilizzato da molte persone, considerando che lo stesso Bitcoin, viene accettato da poche aziende come sistema di pagamento. Dalla sua parte però, Ethereum, permettendo la negoziazione di smart contract, avrà probabilmente più seguito da parte di aziende ed enti.

2. Volatilità: come tutte le criptovalute, anche Ethereum soffre di un'altissima volatilità. Questa porta a forti oscillazioni di prezzo anche nel breve periodo. La volatilià è causata dalla dimensione del mercato delle criptovalute. Essendo piccolo rispetto all'economia mondiale, sono sufficienti piccoli spostamenti di moneta per causare forti oscillazioni sulla quotazione. Quando Ethereum diverrà più utilizzato, la volatilità si abbasserà, e la valuta diverrà più stabile.
3. Sviluppo continuo: Ethereum è un sistema piuttosto recente e quindi in continuo cambiamento. Si dovranno aspettare ulteriori aggiornamenti per avere una rete stabile e matura.[4]

A differenza di Bitcoin, le cui transazioni si basano su output di transazione non spesi, Ethereum si basa sui saldi correnti, chiamati anche transizioni di stato. L'informazione relativa allo stato non è memorizzata nella blockchain, bensì è archiviata in un albero binario nel quale ogni nodo è padre di due figli e il suo hash è dato ricorsivamente dalla concatenazione degli hash dei due blocchi a esso associati. Un portafoglio di criptovaluta memorizza le "chiavi" o "indirizzi" pubblici e privati che possono essere utilizzati per ricevere o spendere Ether. Per indirizzare l'Ether a un conto, è necessario essere in possesso dell'hash calcolato della relativa chiave pubblica.

3.2 Ether e Gas

La sua criptovaluta nativa è chiamata Ether (ETH), essa non viene usata solo come valuta di scambio mediante transazioni, ma anche per l'acquisto di "carburante" necessario all'esecuzione di smart contracts. Ogni transazione comporta l'utilizzo di risorse. Il costo di una determinata attività è chiamato Gas e viene espresso in wei (1 ETH = $1x10^{18}$ wei). [5] Il gas è un token separato dall'Ether, con un proprio mercato in modo da disaccoppiare eventuali fluttuazioni di mercato delle due valute, poiché quest'ultima dipende anche dal grado di utilizzo degli smart contract.
Prima di pubblicare uno smart contract, l'utente deve decidere la quantità massima di carburante che è disposto a consumare e il prezzo che si intende pagare per il gas consumato. Se il carburante è sufficiente per effettuare tutte le operazioni richieste dallo smart contract, ma viene offerto un gas price troppo basso, la transazione rischia di rimanere in sospeso. Se invece il carburante è insufficiente per svolgere le operazioni richieste, il contratto si fermerà e le operazioni non verranno portate a termine, diventando una transazione Out of Gas. Gli ether inviati vengono restituiti, ma il carburante speso viene comunque pagato. Il calcolo di quanto ammonta il pagamento di una transazione, anche se non finalizzata, è il seguente: Gas price * Gas limit. Per una transazione standard, l'importo sufficiente è 21000 gas,

[4] https://www.criptovaluta.it/ethereum/cosa-sono
[5] https://www.criptoguide.it/ethereum-guida-principianti/#Ether_cos8217e_il_gas

mentre per smart contract più complessi l'importo di gas è molto più alto. Il gas, in altre parole, può essere visto come la retribuzione per il lavoro computazionale necessario a inserire la transazione nella blockchain. Dunque, oltre alla quota fissa che il miner riceve per un blocco creato riceverà anche la somma del gas di tutte le transazioni inglobate dal blocco creato.
Una volta pubblicati, gli smart contract vengono effettivamente eseguiti (o "risvegliati") tramite l'esecuzione di particolari transazioni.
Il miner, al momento della selezione delle transazioni da inserire nel proprio blocco, potrà scegliere quante e quali transazioni desidera, rispettando però un limite massimo di gas spendibile per un blocco, attualmente 12.500.000 unità di gas, alzato recentemente da 10.000.000 e facendo pesare un blocco in media dai 35 ai 45 kilobyte. Mediamente, se si conta che una transazione classica costa 21.000 unità di gas, un blocco può contenere quasi 600 transazioni. La media risulta ovviamente più bassa, dato che ultimamente esistono transazioni sempre più complesse e che richiedono più gas per essere portate a termine.
Ether è acquistabile sui principali exchange, scambiabile sui principali siti di scambio criptovalute e trasferibile su wallet dedicati. I wallet, in italiano portafogli, sono in realtà più simili a dei portachiavi: difatti conservano le chiavi private per l'accesso ai propri ether, e la disposizione degli stessi, sulla blockchain di Ethereum.

3.3 Mining

Come spiegato precedentemente, il mining avviene attraverso un algoritmo chiamato Proof of Work. Quello specifico di Ethereum è chiamato Ethash, e utilizza Keccak, una famiglia di funzioni hash di cui fa parte anche la più nota SHA-3. Inoltre viene anche utilizzato un dataset pseudo casuale inizializzato in base alla lunghezza della blockchain, chiamato DAG (Directed Acyclic Graph). Questa DAG, basandosi su un dato non fisso (la lunghezza della blockchain è in continua evoluzione) viene rigenerato ogni 30 mila blocchi, ovvero circa 5 giorni.
Il funzionamento di Ethash, mostrato nella prossima immagine, è il seguente:
Come primo passo, attraverso una funzione Keccak, l'header pre processato del blocco precedente e il nonce generato dal miner, lungo 32 bit, vengono mixati in una sequenza di 128 byte, chiamata Mix 0. Il mix viene poi usato per determinare, con una funzione di fetch, quali dati selezionare dalla DAG. Questi dati estratti e il Mix 0 verranno poi mixati, generando una nuova stringa, sempre di 128 byte, chiamata Mix 1. A questo punto l'operazione di fetch e mix viene ripetuta per 64 volte, fino a raggiungere l'ultima stringa, chiamata Mix 64. Questo risultato verrà infine processato per ottenere una sequenza di soli 32 byte, chiamata Mix Digest, che verrà confrontata con il target del blocco. Se il Mix Digest risulta minore del target, allora il nonce viene considerato corretto e il blocco può essere

convalidato, altrimenti il processo ricomincia con un nuovo nonce generato casualmente dal miner.[6]

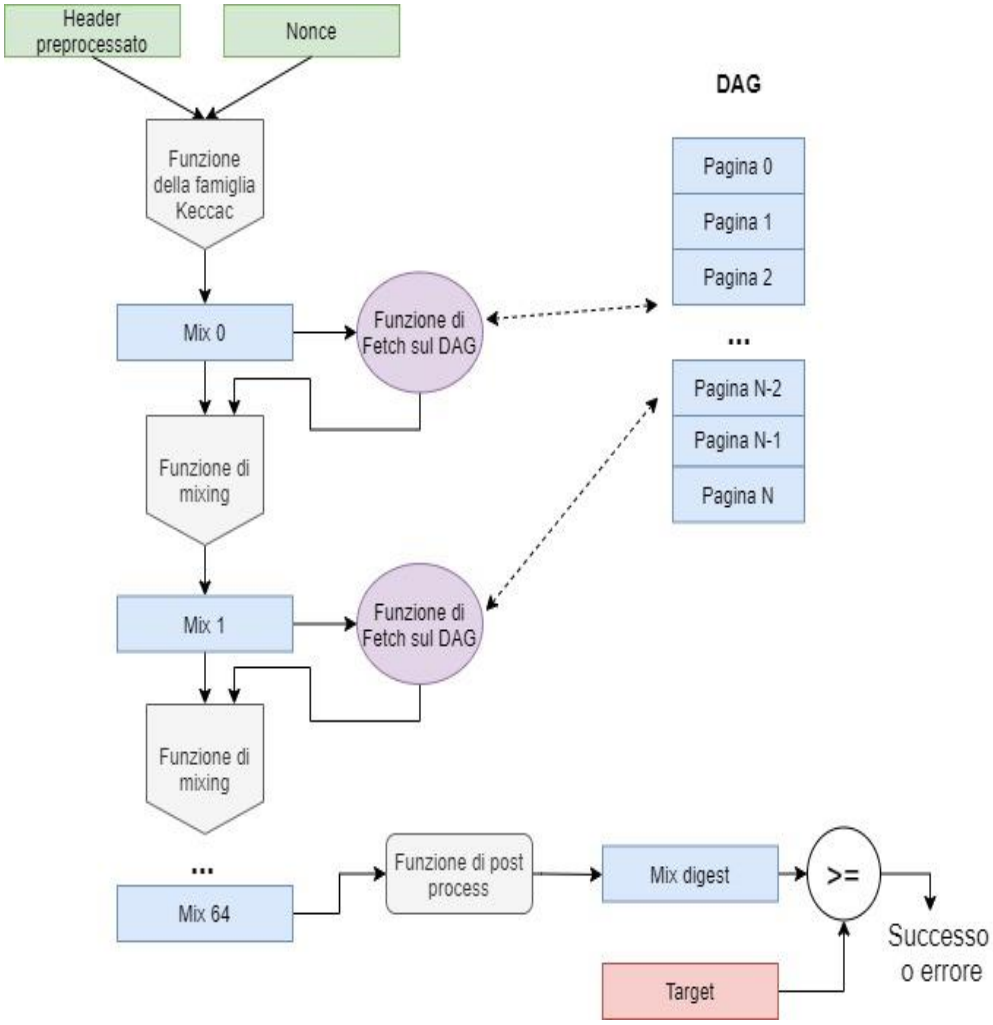


Immagine che mostra il funzionamento dell'algoritmo Ethash, in verde sono evidenziati gli input della funzione, in blu le stringhe intermedie, in grigio le funzioni di hash e di process, in viola le funzioni di fetch che selezionano le pagine del DAG, mentre in rosso il target.

Esistono vari metodi per fare mining su Ethereum: la prima è quella classica, ovvero attraverso il proprio PC. Per ricavare effettivamente qualcosa in termini di guadagno i componenti del pc e la scheda video devono essere scelti con cura, cercando per il primo il minor consumo possibile, la seconda per l'hashrate, ovvero la velocità di hash check effettuabili in un secondo (unità di misura: MH/s megahash al secondo), considerando anche che la DAG ha una dimensione che si avvicina ai 4 GB, quindi una scheda video con meno memoria video risulta inefficace. A partire dal proprio computer, si può decidere di lavorare in "pool" di utenti,

[6] https://cryptonomist.ch/2019/06/15/algoritmi-mining-proof-of-work/

in modo da avere un guadagno minimo ma assicurato, visto che si collabora con milioni di utenti alla ricerca della soluzione, oppure da solo, cosa che risulta molto difficile, soprattutto se si possiede un pc solo. Un'altra alternativa è il cosiddetto "cloud mining". Alcune società permettono di affittare grandi potenze di calcolo per permettere agli utenti di effettuare mining, guadagnando una tariffa fissa in base alla quantità di potenza richiesta dall'utente. Questa opzione risulta più economica rispetto a quella dell'hardware personale, permette di guadagnare meno, ma risolve alcuni problemi del mining classico, tra i quali la produzione di calore, la spesa della bolletta elettrica e permette di evitare problemi che si possono verificare in fase di configurazione della macchina.
Con alcuni aggiornamenti la quota fissa che il miner riceve per ogni blocco minato è stata ridotta. Ad esempio, con l'aggiornamento Bisanzio, il premio è passato da 5 ETH a 3 ETH a partire dal blocco 4.370.000 e l'aggiornamento Costantinopoli ha nuovamente abbassato la ricompensa da 3 ETH a 2 ETH dal blocco 7.280.000.
Durante il mining può succedere che due o più blocchi vengano scelti per essere inseriti nello stesso punto della blockchain, a causa di una risoluzione dalla PoW quasi simultanea o di un ritardo di trasmissione nella rete. A quel punto gli utenti lavoreranno per trovare un nonce che vada bene per uno dei blocchi posti allo stesso livello. Una volta trovato un blocco valido, il padre viene mantenuto nella catena, mentre degli altri, definiti Uncle Blocks, rimane solo l'header, e le transazioni vengono reinserite nella lista di transazioni da processare. A differenza di altre Blockchain, Ethereum ricompensa anche i creatori di questi blocchi scartati, seppur con una ricompensa minore.

3.4 Aggiornamenti della tecnologia Blockchain

Ethereum, essendo una piattaforma innovativa, è anche in continuo aggiornamento. Gli sviluppatori stanno infatti cercando di far evolvere i protocolli della blockchain in una nuova versione, chiamata Serenity. Questo aggiornamento ha come obiettivo quello di migliorare la blockchain sotto cinque principi progettuali: semplicità, resilienza, longevità, sicurezza e decentralizzazione.
Ciò tuttora viene e verrà fatto per gradi, portando poche novità per volta senza sconvolgere completamente il sistema in una sola volta

Il primo di questi aggiornamenti, avvenuto nell'ottobre del 2017, si è occupato di risolvere i problemi di ridimensionamento, implementando due tecniche: il plasma e lo sharding. La prima si è occupata di risolvere i problemi di scalabilità sulla rete di Ethereum, spostando le transazioni degli smart contracts su una catena laterale per ridurre il traffico sulla blockchain principale, dato che queste ultime erano notevolmente aumentate sia di importo che di quantità. Lo sharding invece è un processo di divisione dei dati della catena per evitare congestioni della rete, in modo che ogni nodo debba occuparsi solo di una piccola parte della blockchain, consentendo ad Ethereum di elaborare migliaia di transazioni al secondo, dato che il codice viene eseguito parallelamente e non linearmente.

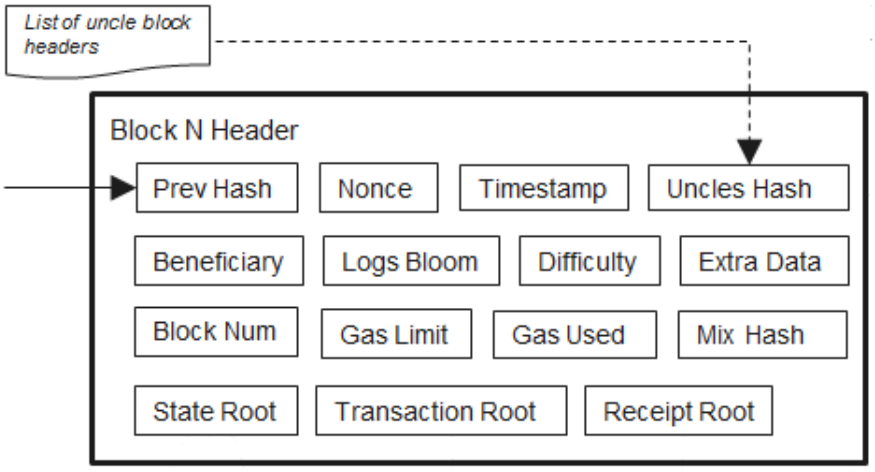
L'altro update, chiamato Costantinopoli e arrivato nell'ottobre del 2019, ha introdotto una serie di aggiornamenti che hanno reso la rete più efficiente e meno costosa in termini di tariffe.
La versione definitiva di Solidity invece dovrebbe arrivare attorno al 2022, ma diversi aggiornamenti e fasi di test sono previste anche per gli anni 2020 e 2021.[7]
Uno dei passi più importanti previsti e ancora in sviluppo è il passaggio dalla Proof of Work alla Proof of Stake. Quest'ultimo è un meccanismo già in uso in alcune blockchain minori e, a differenza della Proof of Work, non si basa più sulla quantità di potenza computazionale che un nodo ha per creare un blocco, ma su un processo di scelta pseudo casuale per selezionare chi sarà il miner del prossimo blocco.
La Proof of Stake funziona nel seguente modo:
gli utenti che vogliono partecipare al processo di creazione del blocco, chiamato non più mining ma forging, devono congelare una certa quantità di criptovalute possedute nella rete (questa somma viene chiamata posta in gioco, o stake). Più è grande la scommessa, più sono le possibilità di vittoria, ma questo non è l'unico fattore che conta. Per fare in modo che non vengano favoriti i più ricchi, vengono usati dei metodi aggiuntivi.
Il randomized Block Selection aumenta le possibilità di estrazione a chi ha una combinazione tra valore hash più basso e stake più grande, mente il Coin Age Selection aumenta la possibilità di estrazione in base a quanto tempo il nodo lascia i propri token come stake, moltiplicando il numero di giorni passati dalla stake con il numero di monete congelate, generando questo indice chiamato coin age. Quando però viene forgiato un blocco, il coin age del creatore viene azzerata e deve aspettare qualche tempo prima di poter essere riselezionato.
Usando questo metodo però, non vengono più generate criptovalute al momento della creazione di un blocco, quindi l'utente che diventa il validatore del blocco otterrà come ricompensa solo le fee delle transazioni nel blocco.
Inoltre ciò rende il sistema sicuro, poiché se qualche nodo prova a forgiare un blocco con transazioni fraudolente, il sistema può confiscare al nodo la sua stake e anche il diritto di poter partecipare all'estrazione per i prossimi blocchi. Quindi, essendo la stake spesso maggiore rispetto alla ricompensa, il nodo truffatore rischia di perdere più di quanto potrebbe guadagnare. Per poter convalidare transazioni fraudolente, un nodo dovrebbe possedere una stake più grande di tutte le altre, situazione anche detta attacco del 51%. Per fare ciò, il nodo dovrebbe possedere il 51% degli token Ethereum in circolazione, cosa quasi impossibile da realizzare.
Un altro motivo per cui questa alternativa è stata ideata è l'evitare il grandissimo consumo elettrico che viene effettuato per svolgere tutti i calcoli, tipico del Proof of Work, permettendo anche a chi non ha un hardware potente di poter partecipare alla sfida per la creazione dei blocchi ed evitare critiche sullo spreco di energia elettrica.[8]

[7] https://etherevolution.eu/la-roadmap-verso-serenity/
[8] https://academy.binance.com/it/blockchain/proof-of-stake-explained

3.5 Struttura del blocco

Il blocco, come già detto in precedenza, è diviso in due parti: il body e l'header.
Il body contiene tutte le transazioni raccolte in quel blocco, mentre l'header contiene informazioni riguardanti il blocco.
L'header di un blocco Ethereum è composto dai seguenti campi:

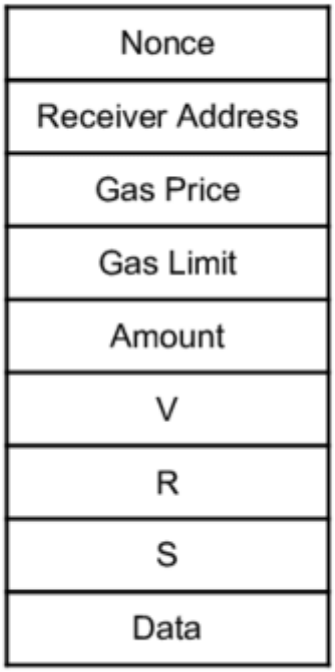
- Prev Hash: hash del blocco precedente
- Nonce: numero generato casualmente dal miner attraverso il quale si ottiene la soluzione della Proof of Work
- Timestamp
- Hash del blocco
- Uncles hash: gli hash dei blocchi Uncles (se ne esistono per quel blocco)
- Beneficiary: indirizzo dell'utente che ha minato il blocco
- Difficulty: un valore corrispondente al livello di difficoltà del blocco, dipende dal timestamp e dalla difficoltà del blocco precedente
- Extra Data: array contenente informazioni aggiuntive
- Block number
- Gas Limit: limite di gas massimo del blocco
- Gas used: gas totale usato dalle transazioni
- Mix hash: hash del nonce



Struttura dell'header

Il body del blocco è composto da diverse transazioni, ognuna delle quali ha la seguente struttura:

- Nonce: contatore che identifica il numero della transazione eseguita dall'utente che l'ha creata
- Receiver Address: Indirizzo del destinatario
- Gas price: prezzo stabilito per unità di gas
- Gas limit: limite di gas che l'utente è disposto a spendere
- Amount: quantità trasferita
- V, R, S: campi usati per firmare la transazione. Possono essere usati per ricavare l'account del mittente
- Data: Corpo contenente dati della transazione.



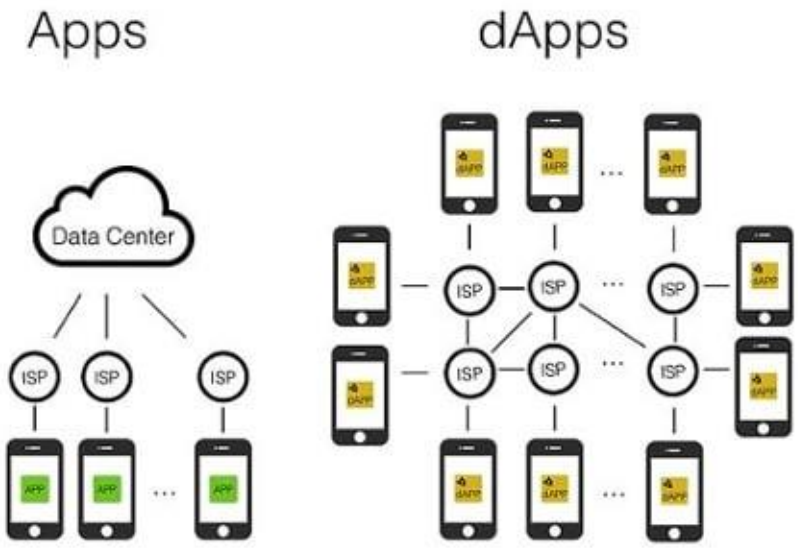
Struttura di una transazione

3.6 DApps

Ethereum è una blockchain programmabile, ciò significa che gli sviluppatori la possono utilizzare per creare delle applicazioni decentralizzate eseguite su una rete p2p di computer, dette dapps, che sfruttano i vantaggi delle criptovalute e della tecnologia blockchain.
Affinchè una dApp possa essere considerata tale, deve soddisfare alcuni requisiti:

1. Dev'essere open-source: deve lavorare autonomamente e non devono esserci entità che controllino la maggioranza dei suoi token. L'applicazione può adattare il suo protocollo in base a miglioramenti proposti, ma dev'esserci il consenso dei suoi utenti per effettuare cambiamenti;
2. Protezione crittografica di dati e operazioni dell'applicazione;
3. Utilizzo di token crittografici per l'accesso;
4. Generazione di token come proof of work, generata attraverso un algoritmo crittografico.

Inoltre Ethereum fornisce una macchina virtuale decentralizzata che permette l'utilizzo degli script, chiamata Ethereum Virtual Machine (EVM).
La caratteristica principale è il codice decentralizzato, ovvero codice ospitato su migliaia di computer ed eseguito successivamente in parallelo. Nel caso sfortunato in cui un nodo della rete vada down o si rifiuti di eseguire il codice, lo stesso codice è eseguito su tutti gli altri computer (nodi) della rete. Le Dapp quindi sono considerate più flessibili, trasparenti, distribuite e resilienti.[9]



Confronto tra applicazioni normali e dApps

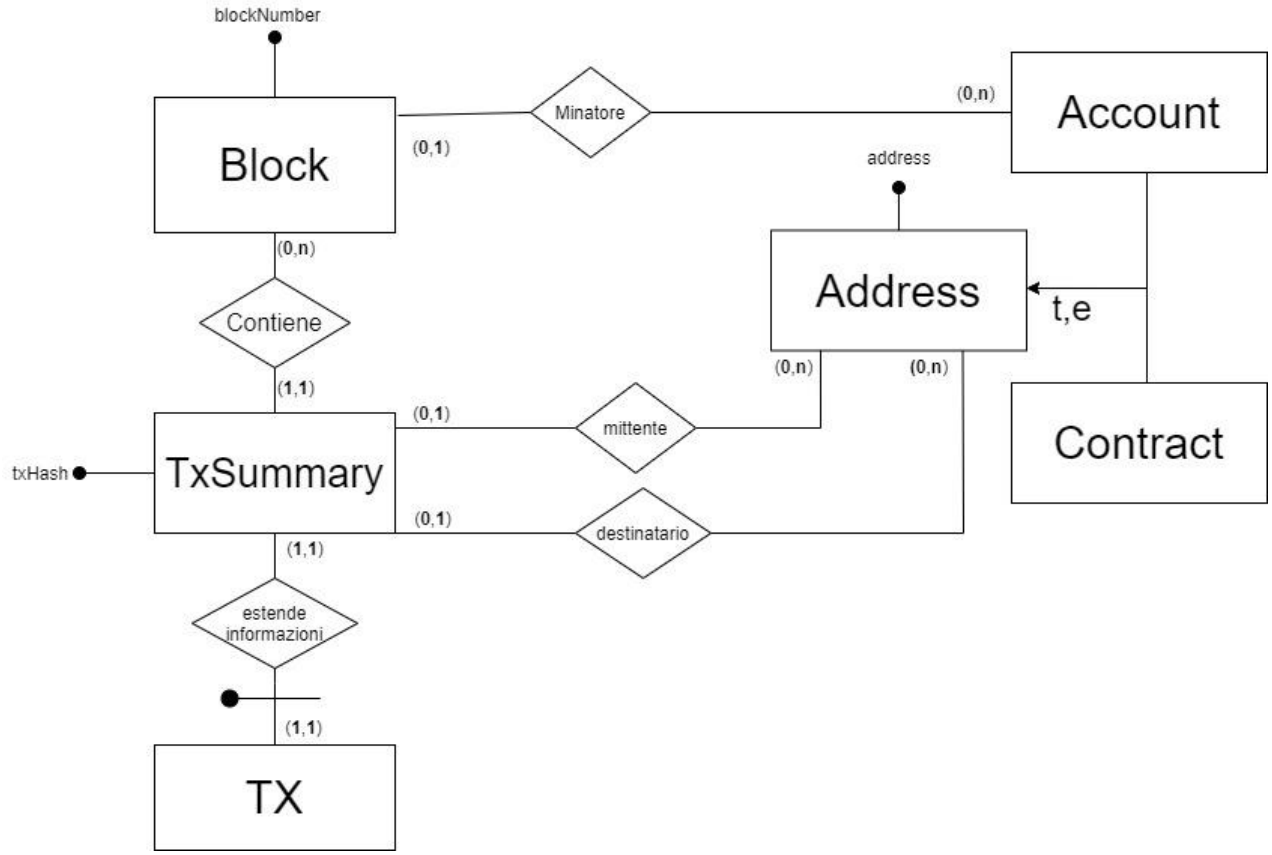
[9] https://cryptonomist.ch/2019/08/17/dapp-cosa-sono-come-funzionano/

4.DATABASE ETHEREUM

Lo scopo del mio tirocinio era quello di creare una base di dati, assieme al mio collega Alberto Baroni, contenente le informazioni estraibili dalla blockchain di Ethereum al fine di renderne più agevole l'analisi e l'integrazione con fonti di dati esterne. Mentre il mio collega si è occupato più degli aspetti legati agli smart contract, io mi sono concentrato sulle informazioni relative alle transazioni. Ciò è stato realizzato attraverso un programma scritto in linguaggio Python e una raccolta di librerie, chiamata web3py, che permette di interfacciarsi con la blockchain e di estrarre diversi dati da essa. I dati vengono scritti in linguaggio SQL su un file che, attraverso la libreria psycopg2, vengono inseriti nella base di dati fornita dall'università situata in un server.

4.1 Modello ER

Nello schema non sono indicati tutti i campi, ma solo le chiavi primarie. Ciò è stato fatto perché sarebbe stato difficile avere uno schema ordinato visto il grande numero di campi di ogni entità.



Modello ER del database

Come si vede nell'immagine, ci sono 5 entità nel database.
La prima, chiamata **Block**, rappresenta i blocchi della blockchain. La sua chiave primaria è il numero del blocco.
La seconda è **TXSummary,** che rappresenta una raccolta di informazioni riguardanti le transazioni e ha come chiave primaria il proprio hash (txhash), mentre la terza, **TX**, raccoglie le informazioni rimanenti e si collega a Summary attraverso txhash. Questa divisione è stata fatta per velocizzare le query, isolando le informazioni più usate in TXSummary e mettendo gli attributi meno usati su TX. TXSummary contiene anche una chiave esportata da Block, ovvero il blocco di appartenenza. Ogni transazione difatti appartiene a uno e un blocco soltanto, ma ogni blocco può contenere al suo interno nessuna o diverse transazioni.
Le altre due entità, **Account** e **Contract**, sono un estensione di un'altra entità, chiamata Address. Ciò è stato fatto perché quando le informazioni di una transazione vengono estratte si hanno come mittente e destinatario due indirizzi, che fungono da chiave primaria per le due entità, che solo in seguito vengono riconosciuti come identificatori di un account o di uno smart contract.
Nell'entità block e nell'entità TXSummary il miner e i due protagonisti delle transazioni possono essere considerabili chiavi esportate da address o, nel caso del miner, precisamente da account. Ciò non è stato fatto nella base di dati perché gli account e i contratti vengono inseriti dopo il blocco e le transazioni e ciò avrebbe portato problemi, perché il blocco e le transazioni non avrebbero trovato la chiave esportata nel database.
Come appena spiegato, ogni blocco è stato creato attraverso il mining da uno e un solo account, ma un account può aver generato più di un blocco, e non è detto che tutti gli account ne abbiano effettivamente generato uno.
Le transazioni hanno un solo mittente e un solo destinatario e queste informazioni sono salvate nell'entità TXSummary. I mittenti e destinatari sono rappresentati da un indirizzo, ma ogni indirizzo può aver compiuto diverse transazioni, così come non averne compiuta nessuna. Un indirizzo può rappresentare sia un contratto che un utente normale, ma solo un utente normale può essere un miner di un blocco.

L'elenco delle tabelle presente nel database è il seguente:

- **Block**: Tabella che rappresenta un blocco della blockchain, i suoi campi sono:
    - blockNumber: rappresenta il numero del blocco;
    - blockGasUsed: quantità di gas usata da tutto il blocco;
    - blockHash: hash identificatore del blocco;
    - blockLogsBloom: campo che permette di cercare più rapidamente una transazione nel blocco;
    - blockNonce: Nonce del blocco;
    - difficulty: campo che indica la difficoltà con cui è stato minato il blocco;
    - extraData: campo che contiene informazioni aggiuntive;
    - gasLimit: limite di gas per il blocco corrente;

    - miner: indirizzo dell'utente che ha minato il blocco;
    - mixHash: hash del Nonce;
    - parentHash: hash del blocco genitore;
    - receiptsRoot : hash del nodo radice della struttura ad albero contenente alcune info riguardanti ogni transazione del blocco;
    - sha3Uncles: SHA3 dei dati di eventuali uncle block;
    - size: dimensione del blocco in bytes;
    - stateRoot: hash del nodo radice di un albero che contiene informazioni sullo stato della blockchain e degli account al momento dell'inserimento del blocco nella catena;
    - timestamp: timestamp del blocco;
    - totalDifficulty: valore che indica la difficoltà dell'intera catena fino al blocco in questione;
    - transactionsRoot: hash del nodo radice di un albero che contiene le transazioni del blocco;
    - uncles: lista degli uncle block;
    - dollarquote: valore in dollari di un ether al momento della creazione del blocco;
    - totalFee: ricompensa totale del blocco;
- **TXSummary**: Tabella che rappresenta una raccolta delle informazioni più importanti relative alle transazioni, e presenta i seguenti campi:
    - txHash: hash univoco della transazione, per questo motivo è la chiave primaria dell'entità;
    - blockNumber: numero del blocco a cui la transazione appartiene, è anche chiave esportata dall'entità block;
    - sender: indirizzo del mittente;
    - nonce: contatore del numero di transazioni effettuato dal mittente;
    - receiver: indirizzo del destinatario;
    - value: importo della transazione, espresso in wei;
- **Tx**: Tabella che raccoglie altre informazioni sulle transazioni. È composta dai seguenti campi:
    - txHash e blockNumber come l'entità Quick (txhash è anche qui chiave primaria);
    - contractAddress: indirizzo di un eventuale contratto usato dalla transazione;
    - cumulativeGasUsed: somma del gas usato dalla transazione corrente e quelle precedenti dello stesso blocco;
    - gas: quantità di gas fornito per poter portare a termine la transazione;
    - gasPrice: prezzo del gas in wei fissato dal mittente per questa transazione;

    - gasUsed: quantità di gas usata dal miner per portare a termine la transazione;
    - input: dati inviati con la transazione;
    - logs: array di oggetti log generati dalla transazione;
    - logsBloom: dati per la ricerca rapida di una transazione;
    - r, s, v: campi usati per firmare la transazione. Possono essere usati per ricavare l'account del mittente
    - status: booleano che indica se la transazione è stata effettuata con successo;
    - transactionIndex: indice della transazione nel blocco;
    - fee: ricompensa ricavata dalla transazione;
- **Account**: Tabella che rappresenta gli account degli utenti, i suoi campi sono i seguenti:
    - Address: indirizzo univoco dell'utente, per questo è anche chiave primaria dell'entità;
    - Balance: bilancio di ether dell'account;
    - txCount: contatore di transazioni effettuate dall'account;
- **Contract**: Tabella che rappresenta gli smart contracts nella blockchain, i suoi campi sono i seguenti:
    - Address: come per gli account, anche gli smart contract hanno un indirizzo univoco, anche in questo caso è chiave primaria;
    - codeSize: diimensione del codice dello Smart Contract;
    - functionNumber: numero di funzioni contenute nello Smart Contract;
    - tokenTotalSupply: bilancio dei token scambiati dal contratto;

4.2 Scelta delle Librerie

Durante il periodo di tirocinio abbiamo testato diverse librerie per l'estrazione di dati dai blocchi tra cui:

- **Web3j**: una collezione di librerie scritte in Java che permettono di lavorare con la rete di Ethereum. Sono molto utili soprattutto per la creazione e la pubblicazione di smart contracts e script, ma poco efficaci per l'estrazione dei dati;
- **Web3js**: librerie scritte in JavaScript, simili a web3j, ma con alcune funzioni più utili ai nostri scopi. Sono state prima accantonate e poi

scartate data la scarsa conoscenza del linguaggio javascript e a seguito dell'utilizzo delle librerie Web3py;

- **Presto**: tool che avrebbe dovuto permettere di eseguire delle query in SQL direttamente sulla blockchain. È stato scartato poiché ci sono stati troppi errori, molti dei quali non sistemabili, in fase di installazione, senza contare la quasi totale assenza di istruzioni su come avrebbe dovuto funzionare;
- **QuickBlocks**: collezione di librerie, tools e programmi scritti per la maggior parte in C++, che permettono di estrarre diverse informazioni dai blocchi. Questa raccolta è risultata abbastanza efficace per l'estrazione e la visualizzazione dei dati, ma poco pratico per trasferirli direttamente in una base di dati, vista anche la non ottima padronanza del linguaggio C++ e la mancanza di librerie in quel linguaggio utili per comunicare con un database;
- **Web3py**: insieme di librerie di Python molto simile alla versione per JavaScript. Questa è stata la libreria usata per scrivere il programma, vista sia la sua completezza nelle funzioni per l'estrazione, sia la buona conoscenza del linguaggio Python e infine la conoscenza delle funzioni principali della libreria psycopg2, che permette la connessione e l'interrogazione di un database PostgreSQL attraverso query scritte in sql.

Oltre a utilizzare quest'ultima serie di librerie, il nostro lavoro si è basato su un programma trovato su GitHub[10], che estraeva alcune informazioni riguardanti i blocchi e le transazioni e le scriveva in un linguaggio SQL simile a quello di PostgreSQL. Dopo aver modificato il programma adattandolo al linguaggio che ci serviva, abbiamo fatto alcuni test facendogli creare un file .sql contenente i blocchi e le transazioni di un certo range di blocchi.
Dopodiché abbiamo fatto in modo che il file appena creato, attraverso la libreria psycopg2, scrivesse tutti i comandi sulla base di dati che ci è stata fornita dall'università. Dopo aver verificato il corretto funzionamento dell'inserimento dei dati, abbiamo messo ulteriormente mano al codice aggiungendo diverse tabelle e attributi alle entità già esistenti.
Inoltre abbiamo aggiunto anche qualche funzione che permetteva di estrarre dati non dalla blockchain (ad esempio il valore dell'ether in base al timestamp) oppure usando altre API (bilancio dei token).

[10] https://github.com/validitylabs/EthereumDB

4.3 Funzionamento del programma

Il programma si basa su diversi file scritti in python. Il file principale è database.py.
La prima azione effetuata nel programma è collegare la libreria web3 alla rete della blockchain, ciò si può fare attraverso un account Infura, che permette all'utente di collegarsi a un altro host che ha già salvata la propria copia della blockchain. In alternativa, modificando il comando, si può far collegare il programma a una copia fisica della blockchain che si possiede sul dispositivo.

```
# 1. connection via Infura
web3 = Web3(Web3.HTTPProvider("https://mainnet.infura.io/v3/6484d4add39247bb8a0da7e8ae5ce08a"))

# 2. or connection via local node
#web3 = Web3(Web3.IPCProvider('/your-path-to/geth.ipc'))
```

Poi viene impostata la quantità di blocchi che vogliono essere caricati sul database e viene deciso il blocco di partenza andando a verificare qual è stato l'ultimo caricato nel file lastblock.txt. Se il file è vuoto o non esiste, allora il programma comincerà a estrarre i dati dal primo blocco della blockchain. Inoltre viene creato, o se esiste già svuotato, un file SQL sul quale verranno scritti tutti i comandi PostgreSQL che devono essere inseriti del database e, nel caso in cui il database risulti vuoto, vengono aggiunti i comandi di creazione delle tabelle e degli indici. Viene utilizzato un file intermedio così da poter analizzare il problema in caso di errore e per avere un punto di ripristino preciso in caso di errore di un comando. Nel caso in cui il programma si interrompesse su una transazione, se non usassi questo metodo avrei un blocco caricato in memoria con solo alcune delle sue transazioni, invece così il blocco non vienecaricato.

```
#numero dei blocchi che volgio caricare
Nblocks =  10000
start_time = time.time()
#apro lastblock.txt per vedere da quale blocco partire , altrimenti parto dal blocco start
try:
    with open('lastblock.txt', 'r') as f:
        start = int(f.read())+1
except FileNotFoundError:
    start =  0

#se il Database non esiste lo creo e lo inizializzo
file_name="database.sql"
file_new= not os.path.exists(file_name)

if file_new:
    print("creazione sql")
    create_sql()
else:
   file1 = open("database.sql","r+")
   file1.truncate(0)
   file1.close()
```

Il programma inizia il primo ciclo, all'interno del quale estrae i dati dell' n-esimo blocco in un dizionario, poi richiama una funzione che scrive sul file i dati in formato SQL. Dopodichè, prima di iniziare il ciclo innestato, il programma tiene conto del numero del blocco per segnare la quota fissa che il miner riceve (quota che, come detto in precedenza, si è abbassata nel corso degli anni) e inserisce, nel caso in cui non sia già all'interno del database, anche l'account del miner di quel blocco. Il ciclo interno itera su ogni transazione presente nel blocco, aggiungendo sia i dati della transazione che i dati del mittente e del destinatario al solito file di testo. Alla fine di questo ciclo viene aggiunta la ricompensa totale ricevuta dal minatore nella tabella blocco, sommando al valore totalFee per ogni transazione il prodotto tra il gas usato e il prezzo di quest'ultimo.

```
count = 0
#loop over all blocks
for block in range(start, start+Nblocks):

    #scrivo il blocco nel database
    block_table, block_data = order_table_block(block,web3)
    file1 = open("database.sql","a")
    file1.write(replace_wordb(block_table))
    file1.write(replace_worda(block_table,web3,"miner"))

    #calcolo il valore statico di ricompensa
    totFee = 0
    if block < 4370000 :
        totFee = 5
    elif block < 7280000 :
        totFee = 3
    else :
        totFee = 2
    #itero su ogni transazione del blocco
    for hashh in block_data['transactions']:
        quick_table, tx_data = order_table_quick(hashh,block, web3)

        #list of tx data that will go to the DB
        TX_table = order_table_tx(tx_data,hashh, web3)
        #scrivo nel file ogni transazione sulla tabella quick e tx
        #e ogni account sulla tabella account
        file1.write(replace_wordt(TX_table))
        file1.write(replace_wordq(quick_table))

        #sommo le fee da scrivere nel blocco
        totFee = totFee + web3.fromWei((TX_table.get("gasUsed") * TX_table.get("gasPrice")),'ether')

        file1.write(replace_worda(quick_table,web3,"from"))
        file1.write(replace_worda(quick_table,web3,"to"))

    #aggiungo fee totali al blocco
    file1.write(replace_wordFeeBlock(totFee,block))
    file1.close()

    count = count + 1
    print(count)
    #dump output every 2 blocks
```

Ogni 100 blocchi il programma apre una connessione con la base di dati attraverso la libreria psycopg, inserendo i dati estratti fino a quel momento nel file database.sql. Dopodiché il file viene svuotato e viene presa traccia del numero dell'ultimo blocco caricato in lastblock.txt, così in caso di problemi il programma sa da quale blocco può ripartire. Una volta terminato il ciclo esterno il programma si collega per l'ultima volta alla base di dati e inserisce il contenuto rimanente del file e viene aggiornato il contatore dell'ultimo blocco analizzato.

```
#update the current block number to a file
if (count % 1000) == 0:
    connection = psycopg2.connect(user = "deboni", password = "eth2004", host = "localhost" , port = "5437 ", database = "ethdb")
    with connection:
        with connection.cursor() as cursore:
            cursore.execute(open("database.sql", "r").read())
    connection.close()
    with open('lastblock.txt', 'w') as f:
        f.write("%d" % block)
    file1 = open("database.sql","r+")
    file1.truncate(0)
    file1.close()
```

Il file organizer.py è sostanzialmente una libreria di funzioni che estraggono i dati dalla blockchain e li trasferiscono in un dizionario. Questo file non è stato modificato, poiché le funzioni di questo file non hanno mai causato problemi.

Il file sql_helper è una raccolta di funzioni che trasferiscono e traducono i dati dal dizionario al linguaggio SQL, è stato modificato, perché conteneva alcuni errori nella sintassi SQL e sono state aggiunte altre funzioni inizialmente non comprese.

- **Create_sql**: funzione che inizializza il file SQL, includendo la creazione di tutte le tabelle e degli indici. Qua sono stati aggiunti i comandi di creazione delle tabelle che non erano previste originariamente e modificati quelli già presenti con l'aggiunta dei campi inizialmente non compresi, come dollarquote e totalFee per il campo block.
- **Replace_wordb**: funzione che estrae i dati dal dizionario del blocco e scrive il comando per inserire il blocco nel database. Prima di scrivere il comando, vengono tolti i caratteri apice dal campo uncles (il carattere crea diversi problemi, perché in SQL anche il singolo apice può essere interpretato come chiusura di una stringa) e viene modificato il formato del timestamp. Per scrivere la stringa viene sfruttato la logica di sostituire il campo del dizionario con il contenuto del campo stesso; per fare ciò il nome del campo dev'essere delimitato dal carattere spazio. Quindi i nomi dei campi

hanno degli spazi che, una volta sostituiti dai dati, verranno tolti facendo spazio agli apici. Inoltre viene anche richiamata un'altra funzione, chiamata getPriceAtDay, che aggiunge il valore della criptovaluta al momento della creazione del blocco.

- **Replace_wordt**: funzione simile alla precedente, che estrae i dati dal dizionario della transazione e scrive il comando per inserire il blocco nel database. Anche qua viene sfruttata la sostituzione dei campi del dizionario nella stringa con i loro valori corrispondenti nel dizionario e anche qui da un campo, in questo caso logs, vengono tolti gli apici.
- **Replace_wordq**: funzione molto simile alla precedente, poiché fa la stessa cosa della precedente, ma scrivendo il comando per la tabella quick, che raccoglie le informazioni principali di una transazione.
- **insertAccount**: funzione che scrive il comando per inserire gli account nella base di dati. Attraverso una funzione di web3 il sottoprogramma capisce dall'indirizzo se si parla di un account normale o di uno smart contract, e in base al risultato verrà richiamata una sottofunzione che inserirà la stringa corretta.

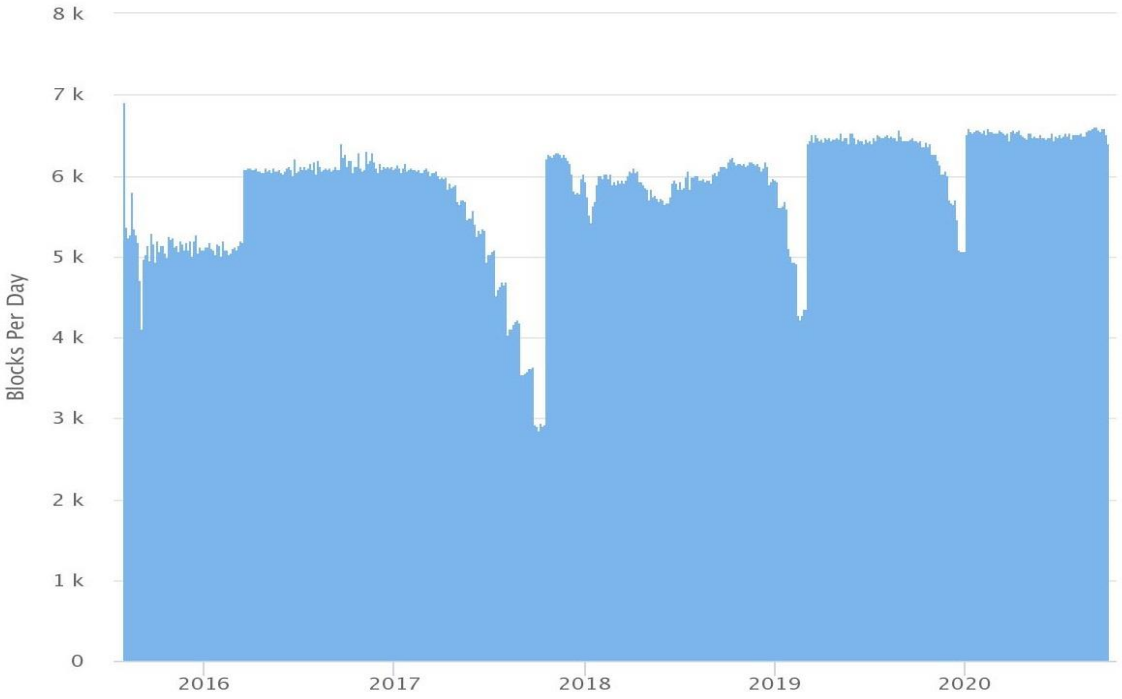
5.STATISTICHE SULLA BLOCKCHAIN

5.1 Statistiche sui blocchi

La prima statistica che voglio analizzare è il numero di blocchi creati al giorno. La query con cui può essere estratta questa informazione è la seguente:

```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , COUNT(*)
FROM Block B
GROUP BY data_mercato;
```

dove data_mercato è il campo timestamp convertito in una data di tipo GG MM AAAA.
Il grafico che ne risulta è il seguente:

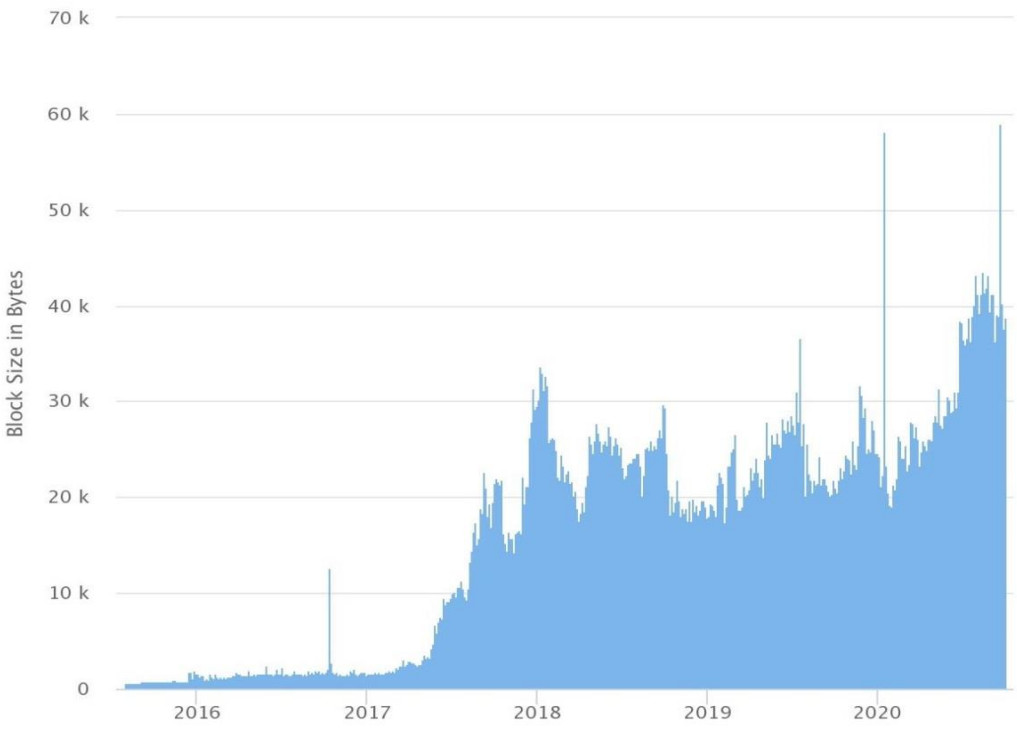


Si può notare come il numero di blocchi sia rimasto più o meno costante per tutto il tempo, ad eccezione di alcuni cali. Il calo più importante si vede attorno ad ottobre del 2017. Questo perché, già dal 2015, i creatori della blockchain erano convinti che il Proof of Stake fosse la scelta giusta per la

creazione dei blocchi, e speravano di riuscire a convertirla in un paio d'anni. Quindi i progrmamatori decisero di modificare il codice del Proof of Work, creando una crescita esponenziale nella difficoltà di mining. Questo evento è noto come "Difficulty Bomb" e si è iniziato a percepire già verso la fine del 2016. Nell'ottobre del 2017, quando il tempo di mining era ormai più che raddoppiato (era arrivato a 30 secondi per blocco), il team di Ethereum ha deciso di resettare il timer della "Difficulty Bomb", dato che la blockchain non era ancora pronta per la conversione del sistema di creazione blocchi. Gli altri due cali che si vedono sul grafico sono dovuti allo stesso motivo di quello di ottobre 2017, perché il calo dopo il riavvio del timer avviene circa dopo 12 mesi.

Il grafico successivo mostra la media giornaliera della dimensione dei blocchi, con i dati raccolti attraverso la seguente query:

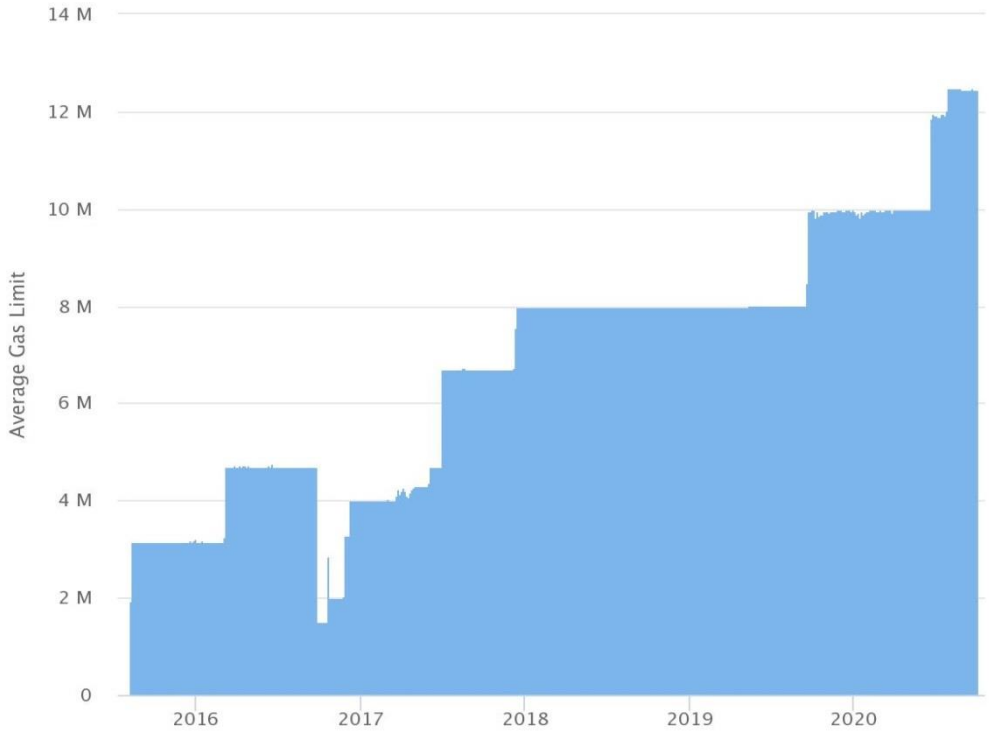
```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , B.size
FROM Block B
GROUP BY data_mercato;
```



La dimensione dei blocchi è aumentata notevolmente nel tempo, e le cause sono principalmente due: l'aumento del gas limit nel tempo e l'aumento degli utenti, e quindi delle transazioni, che vedremo in seguito.

Il grafico seguente mostra il gas limit giornaliero, la query con cui si può ricavare è la seguente:

```
  SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , B.gasLimit
FROM Block B
GROUP BY data_mercato;
```



Dal grafico si può notare che il limite di gas per blocco continua a crescere. Questo limite può essere aumentato attraverso una votazione dei miner, che possono decidere di aumentare il gas limit di 1/1024 per ogni blocco creato. Questa manovra rischia di intaccare la decentralizzazione della rete se usata male, visto che con l'aumento della dimensione dei blocchi serviranno dispositivi sempre più capienti e potenti, senza contare che potrebbe anche portare a problemi di propagazione dei blocchi. Il vantaggio però, per i miner, è quello di poter inserire più transazioni (o comunque transazioni più complesse e quindi più onerose) in un blocco solo.

5.2 Statistiche sulla valuta

Il valore giornaliero dell'ether si può ricavare con un'altra semplice query:

```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , B.gasLimit
FROM Block B
GROUP BY data_mercato;
```

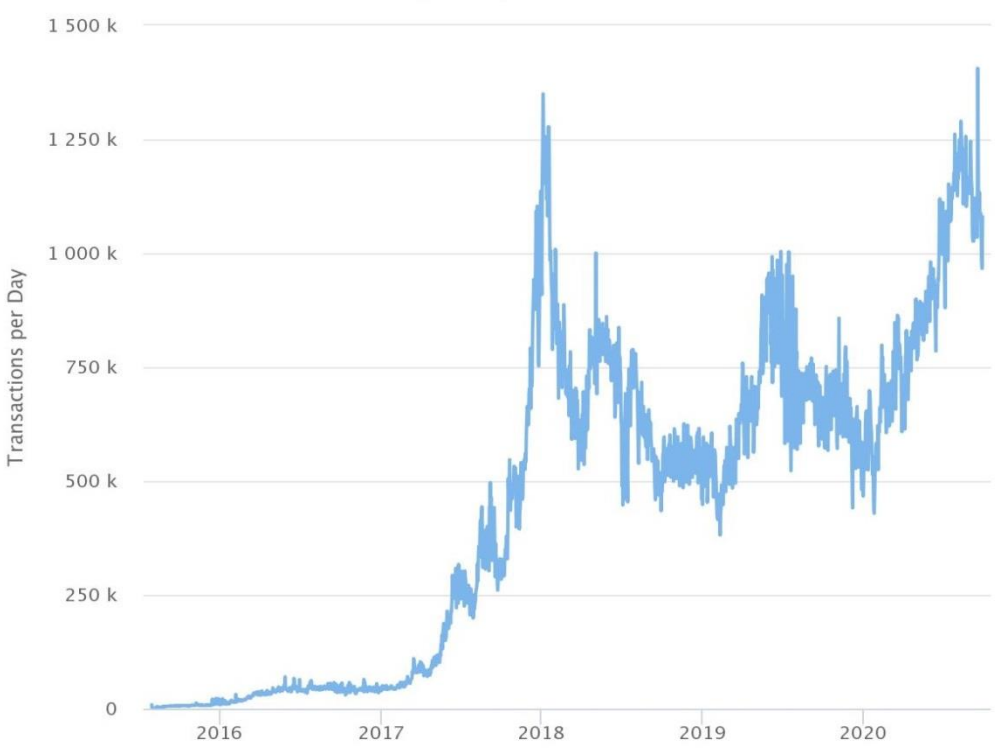
Da questo grafico si può notare come il valore di questa criptovaluta sia altamente volatile e instabile. Fino al 2017 il valore dell'Ether si aggirava sui 10 dollari, cominciando a salire nei primi mesi di quell'anno, fino al picco di circa 1300 dollari nel gennaio del 2018, per poi scendere nuovamente e risalire a 700 dollari nel maggio dello stesso anno. Da metà 2018 fino a inizio 2020 il valore si è mantenuto tra i 100 e i 250 dollari, per poi risalire negli ultimi mesi con un picco di 500 dollari. Il valore attuale si aggira intorno ai 350 dollari.
Un'osservazione che si può fare confrontando questo grafico con quelli di altre blockchain (ad esempio Bitcoin e Litecoin) è che molte blockchain hanno avuto un picco nello stesso periodo. Essendo il picco a circa 2 mesi di distanza di quello di Bitcoin, si può dedurre che il successo di Ethereum, ma anche di altre criptovalute, sia dovuto anche all'espansione di Bitcoin e al concetto di criptovalute stesso.
Tra le varie cause di questo sviluppo, oltre alla crescita di notorietà, si è parlato anche di una manipolazione del mercato da uno speculatore, che, attraverso molte transazioni da milioni di dollari eseguite con un digital token con valore stabile di un dollaro a unità, avrebbe portato a questo aumento di valore delle criptovalute.[11]

[11] https://leganerd.com/2019/11/05/la-crescita-vertiginosa-del-bitcoin-nel-2017-e-stata-pilotata-da-un-unico-speculatore/

5.3 Statistiche sulle transazioni

Il grafico delle transazioni può essere ricavato attraverso la seguente query:

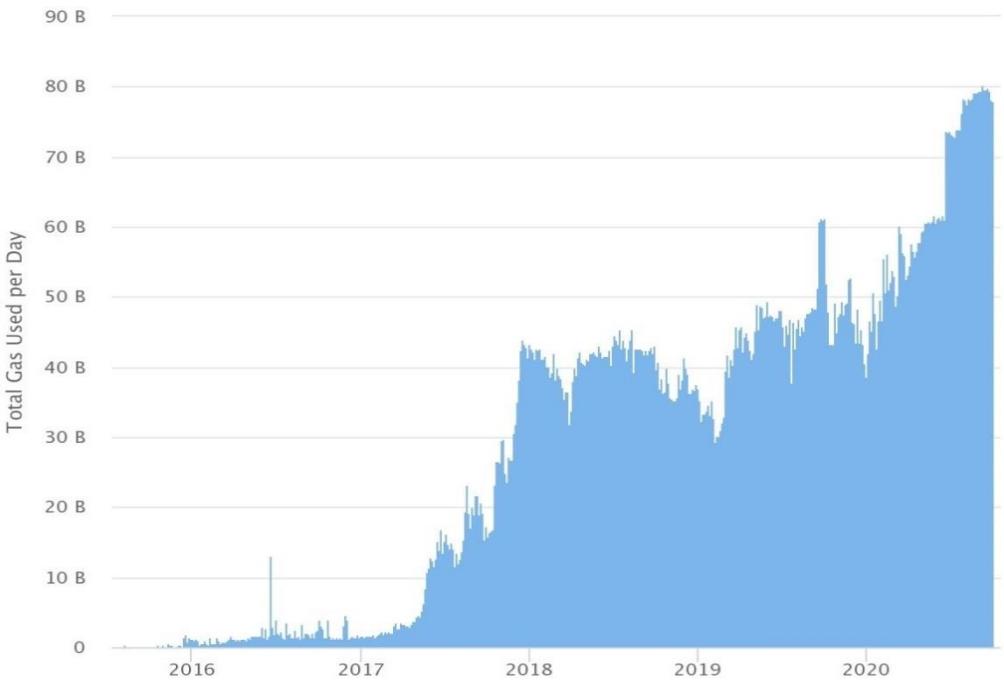
```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , COUNT(txhash)
FROM Block B
JOIN TXSummary S ON B.blockNumber = S.blockNumber
GROUP BY data_mercato;
```



Come detto in precedenza, il picco delle transazioni è avvenuto nello stesso momento del picco del valore dell'ether, e con questo di conseguenza anche l'aumento dell'utenza, che facendo più transazioni ha portato il valore dell'ether ad alzarsi. Nonostante l'aumento del gas limit dei blocchi e della loro grandezza in memoria, il numero di transazioni non è sempre aumentato. Ciò è dovuto agli smart contract, che occupano molto più spazio di una semplice transazione, ma portano anche più introiti al minatore che riesce a includerli nel proprio blocco. Ciò è dovuto al fatto che il numero decisamente più alto di funzioni in uno smart contract rispetto a una normale transazione implichi più azioni da portare a termine dal minatore, e quindi una maggior richiesta di gas per completare le transazioni inerenti ai contratti.

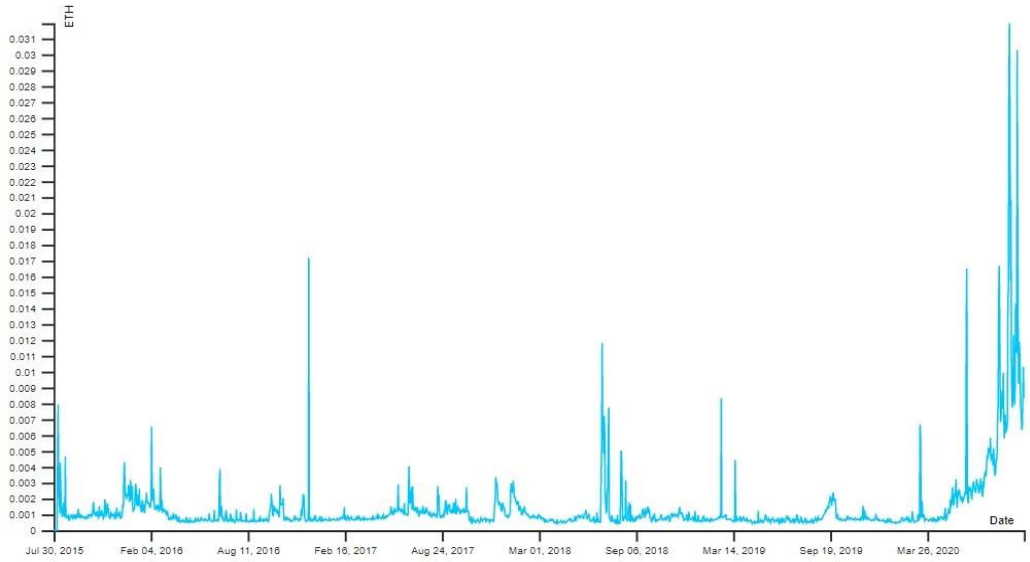
Per quanto riguarda il gas usato, di seguito vengono mostrati query e grafico riguardanti il consumo giornaliero di gas:

```
SELECT  (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , SUM(blockGasUsed)
FROM Block B
GROUP BY data_mercato;
```



L'aumento del gas è dovuto anche all'aumento del gas limit per blocco citato in precedenza.
Il prossimo grafico e la prossima query mostrano il valore medio giornaliero delle ricompense generate dalle transazioni:

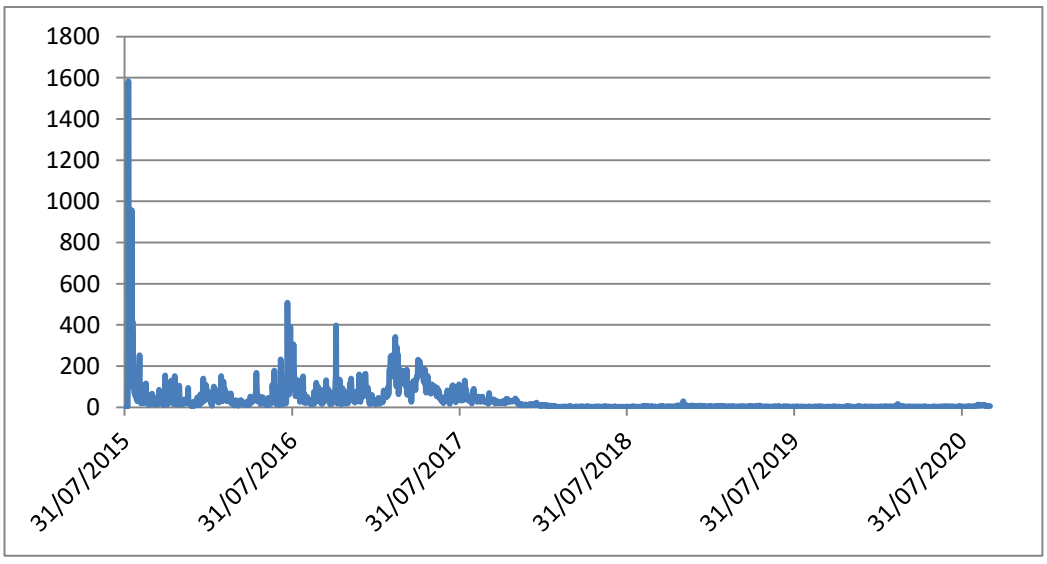
```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , AVG(T.fee)/10^18
FROM tx T
JOIN block B ON T.blocknumber = B.blocknumber
GROUP BY data_mercato;
```



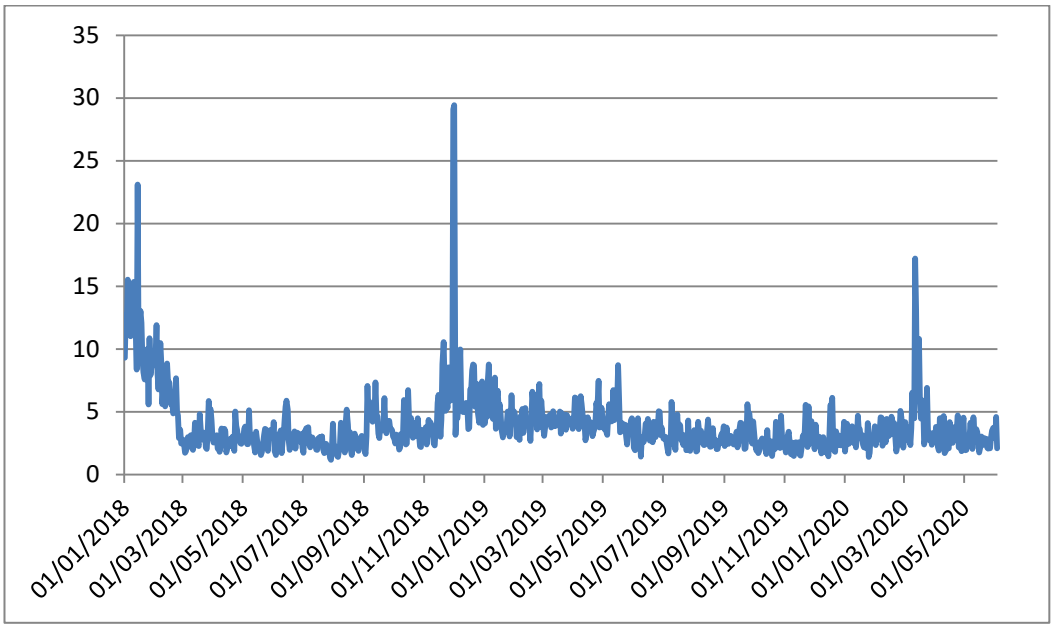
Da questo grafico si può notare come in realtà la ricompensa media per una transazione sia davvero bassa, a indicare che molte di esse non utilizzano smart contract o perlomeno ne utilizzano alcuni poco onerosi. Nell'ultimo periodo c'è stato però un costante aumento del valore medio di ricompensa, sintomo che le transazioni stanno diventando sempre più complesse.

L'ultima statistica riguarda la quantità media di criptovalute spostate al giorno per transazione:

```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , AVG(S.value)/10^18
FROM TXSummary S
JOIN block B ON S.blocknumber = B.blocknumber
GROUP BY data_mercato;
```



```
SELECT (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE as data_mercato , AVG(S.value)/10^18
FROM TXSummary S
JOIN block B ON S.blocknumber = B.blocknumber
WHERE (TIMESTAMP 'epoch' + timestamp * INTERVAL '1
second')::DATE >= '2018-01-01'
GROUP BY data_mercato;
```



Zoom del grafico usando le statistiche dal 2018

Da questo grafico si può notare come il valore di ether trasferiti si stia tendendo a stabilizzare in una cifra molto bassa, che si aggira tra i 2 e i 5 ether per transazione. Ciò probabilmente è dovuto al fatto che a partire dal 2018 il valore dell'ether è cresciuto esponenzialmente rispetto a prima, anche se con diversi picchi verso il basso, ma mantenendosi comunque nettamente più alto.

6.BIBLIOGRAFIA E SITOGRAFIA

- Ethereum yellow paper, DR. Gavin Good (https://ethereum.github.io/yellowpaper/paper.pdf)
- Ethereum For Dummies, A. Wiley Brand

- https://etherscan.io/
- https://blog.osservatori.net/it_it/distributed-ledger-technology-significato
- https://www.trend-online.com/blockchain-bitcoin.html
- https://it.wikipedia.org/wiki/Blockchain
- http://www.mokabyte.it/2018/03/realblockchain-3/
- https://en.bitcoin.it/wiki/
- https://academy.binance.com/it/articles/blockchain-oracles-explained
- https://medium.com/@cryptojj8/orphaned-and-uncle-blocks-in-blockchain-6f7d953a4a50
- https://ethereum.org/it/what-is-ethereum/
- https://www.blockchain4innovation.it/criptovalute/andamento/cose-quali-gli-ambiti-applicativi-ethereum/
- https://www.criptovaluta.it/ethereum/cosa-sono
- https://www.criptoguide.it/ethereum-guida-principianti/#Ether_cos8217e_il_gas
- https://etherevolution.eu/transazione-ethereum-out-of-gas/
- https://ethgasstation.info/blog/gas-limit/
- https://etherevolution.eu/la-roadmap-verso-serenity/
- https://etherevolution.eu/ethereum-si-aggiorna-costantinopoli-in-arrivo/
- https://cryptonomist.ch/2019/08/17/dapp-cosa-sono-come-funzionano/
- https://www.tokens24.com/it/cryptopedia/
  https://medium.com/@fabiopezzotti/cosa-sono-le-dapps-9e52c1057cf
- https://www.criptomane.com/mining/ethereum/#COME_FUNZIONA_IL_MINING_DI_ETHEREUM
- https://cryptonomist.ch/2019/06/15/algoritmi-mining-proof-of-work/
- https://etherscan.io/
- https://ethgasstation.info/blog/ethereum-block-size
- https://academy.binance.com/it/blockchain/proof-of-stake-explained

- https://www.criptoinvestire.com/come-funziona-la-crittografia-nelle-blockchain.html
- https://medium.com/validitylabs/how-to-interact-with-the-ethereum-blockchain-and-create-a-database-with-python-and-sql-3dcbd579b3c0
- https://spectrum.ieee.org/computing/networks/ethereum-plans-to-cut-its-absurd-energy-consumption-by-99-percent
- https://leganerd.com/2019/11/05/la-crescita-vertiginosa-del-bitcoin-nel-2017-e-stata-pilotata-da-un-unico-speculatore
- https://blockchair.com/