Thesis for Bachelor’s Degree

The chart page development

for GIST SWAP using The Graph

The chart page development for GIST SWAP using The Graph

x

MinGook Kim

School of Mechanical Engineering

Gwangju Institute of Science and Technology

2022

2022

;ㅣㅏㅓㅗ호ㅕㅠ효

MinGook Kim

The chart page development

for GIST SWAP using The Graph

The Graph를 활용한

GIST SWAP 차트 페이지 개발

The chart page development

For GIST SWAP using The Graph

Advisor : Professor Jae Wook Lee

Co-Advisor : Professor Heung No Lee

by

Min Gook Kim

Department of Mechanical Engineering

Gwangju Institute of Science and Technology

A thesis submitted to the faculty of the Gwangju Institute of Science and Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Mechanical Engineering

Gwangju, Republic of Korea

2022. 06 . 07 .

Approved by

Professor Jae Wook Lee

Committee Chair

The chart page development

For GIST SWAP using The Graph

Min Gook Kim

Accepted in partial fulfillment of the requirements for the degree of Bachelor of Science

June 7, 2022

Committee Chair

Prof. Jae Wook Lee

Committee Member

Prof. Heung No Lee

|  |  |
| --- | --- |
| MS/ME  20175071 | MinGook Kim (김민국). The chart page developmentFor GIST SWAP using The Graph (The Graph를 활용한 GIST SWAP 차트 페이지 개발). Department of Machenical Engineering. 2022. 30p. Prof. Jae Wook Lee |

**Abstract**

    Blockchain technology has enabled e-commerce between individuals without third-party intermediaries. As the blockchain ecosystem grows, many users are entering the ecosystem. In this trend, interest in the task of creating a fast speed of writing and reading of the Ethereum blockchain is growing to increase the user experience. Among them, blockchain data reading technology is not indexed due to the nature of the blockchain in which data of all DApps are stored in one place, so it cannot be simply inquired unlike databases of ordinary server client-type services. To solve this reading problem, many projects are attempting to index blockchain data separately so that it can be viewed simply like the existing method. Among them, The Graph, a decentralized blockchain data indexing platform used by Uniswap, one of the most successful Defi projects, is one of the successful cases.

    In this paper, we analyze the ecosystem of The Graph Network and analyze which logic allows for decentralization and infinite scalability at the same time. In addition, subgraphs are deployed to The Graph to index data from Uniswap, one of the DApps existing on the Ethereum blockchain platform. After that, we use indexed subgraphs to query Uniswap's data to lay the foundation for the subgraph of GIST SWAP, Uniswap's copy project.

**Contents**

**Abstract ……………………………………………………………………………………………………… i**

**List of Contents ……………………………………………………………………………………………… ii**

**List of Figures ……………………………………………………………………………………………….. ⅲ**

**I. ITRODUCTION ………………………….…………………………………….…………………………… 1**

**II. Body**

**2.1. Ecosystem of Graph Network …………………………………….…………………………… 2**

**2.1.1. What is The Graph? …………………………………………….………………… 2**

**2.1.2. How The Graph works? …………………………………………………………... 3**

**2.1.3. Ecosystem participants .………………………………….……………………… 13**

**2.1.4. Token Economics ………………………………………….……………………… 13**

**2.2. Graph Network protocol analysis …………………………………….……………………… 13**

**2.2.1. How is infinite expansion possible? ……………………….……………………… 13**

**2.2.2. How The Graph store datas? …………………………….……………………… 13**

**2.3. Indxing data using The Graph ….…………………………………….……………………… 13**

**2.3.1. Subgraph Structure & Preparation files ……………….……………………… 13**

**2.3.2. Uniswap v3 subgraph depoly …………………………….……………………… 13**

**2.3.3. messari/subgraphs/uniswap-v3 depoly ………………….……………………… 13**

**2.3.4. Uniswap v3 tokens, TVL value extraction ……………….……………………… 13**

**Ⅲ. Conclution**

**IV. Reference**

**List of Figures**

Figure 1. Data structure of an Ethereum transaction 3

Figure 2. The data flow after deployed subgraph [4] 4

Figure 3. subgraph entity example 7

Figure 4. Type error when deploy the Uniswap v3 subgraph 9

Figure 5. i32 error code 10

Figure 6. fix i32 error code 10

Figure 7. deploy complete on The Graph 11

Figure 8. playground log error 11

Figure 9. indexing error on playground 12

Figure 10. complete subgraph that deploy and sync data successfully 13

Figure 11. Requested query using deployed subgraphs 14

Figure 12. Uniswap v3 Total Value Locked value on Optimism network [17] 14

**Ⅰ. Introduction**

    Blockchain technology has created a system that cannot falsify data, allowing reliable safe transactions and data processing without a reliable third-party intermediary. Countless people who sympathized with this entered the blockchain ecosystem and numerous DApps were developed. Among them, successful DApps such as Uniswap came out, and the frequency of DApps reading data from the blockchain increased as more users increased.

    However, blockchain does not have one centralized server for one app like servers in the existing IT industry. For example, DApps running on Ethereum all store their own data in the same Ethereum. If you want to get any data from DApp through filtering, you can retrieve information without a big problem because the data structure is indexed well, but blockchain is a database that stores all information from DApps, so you have to search for all transactions and extract metadata through hash values. As blockchain develops more in the future, platforms like Ethereum will have more users, and querying data will take longer as the amount of data to filter increases.

    To solve this problem, DApp can constantly update blockchain data by building its own indexing server, store the necessary information in the database, and when DApp users request a query, it can solve the above speed problem by sending indexed data from its own server. However, if the query results are processed in this way, administrative costs will be spent on continuously expanding the server as the number of users increases. In addition, this is an inappropriate method because both decentralization and security, which are important elements of the blockchain, are missed.

    Therefore, creating a blockchain indexing system that is a decentralized method and can extend servers indefinitely without special management as the number of users increases is one of the key challenges, which The Graph addresses.

    Uniswap is one of the successful Defi projects, with total trading volume exceeding 1 trillion dollar [1]. As transactions occur, the token exchange fee changes in real time and the ability to quickly check the information in charts is essential, so the Uniswap team uses The Graph to index the information so that users can see the transaction information in almost real time.

    GIST SWAP, an in-school Defi project, is a task that implements the function of Uniswap as a copy project of Uniswap. GIST SWAP is also a Defi project, and as it grows, querying price information becomes very important. Despite the powerful function of The Graph, information is scattered in many places, so it is not easy to understand The Graph immediately and start indexing.

    Therefore, in this study, we will start with what The Graph is, explain how it works, and who the participants are, and describe what ecosystem the Graph Network has. It will also describe how this technology has fundamentally decentralized beyond the ecosystem and has become a scalable solution. Finally, we will actually use The Graph to query Uniswap's data and explain how to write code to index blockchain data. Through this study, it is expected that entry barriers to entering the The Graph ecosystem and indexing blockchain data can be lowered.

**Ⅱ. Body**

**2. 1. Ecosystem of graph network**

**2. 1. 1. What is The Graph?**

    The discourse held by most scalability-oriented blockchains is about writing and is often measured by the number of transaction processes per second (TPS). Ethereum is 15 to 30 TPS, Binance smart chain is 160 TPS, and Solana is 50,000 TPS. Investors have invested hundreds of millions of dollars for the scalability of the blockchain's writing ability [2]. However, the more write events occur on the blockchain, the more read events occur. Taking SNS as an example, if a writing event occurs once by a person with 1,000 followers, the action of reading the writing information occurs 1,000 times. As the number of DApp users increases, the number of users who generate writes will continue to increase, and the number of events that read the information will increase exponentially. Therefore, securing speed for read processing for a pleasant DApp environment is a very important problem.

    The reason why there is a problem with the read processing speed of the blockchain is that the blockchain has a single table.

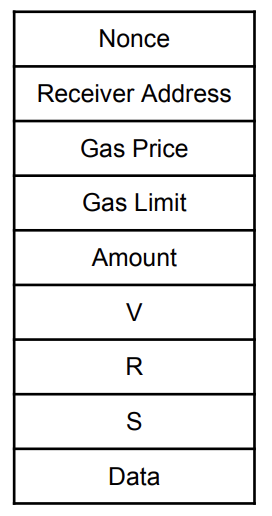


Figure 1. Data structure of an Ethereum transaction

    Figure 1 describes the structure of Ethereum transactions, and event information in all DApps is stored in the above format. For example, information such as the Total Value Locked (TVL) value, which identifies 10 Uniswap tokens in ascending order by name, is not defined in Ethereum. Therefore, in order to obtain the value, it is necessary to aggregate the values obtained from metadata using the IPFS hash of the events that have occurred so far. Despite these simple queries, if not considered in the initial design of Smart Contract, DApp can take hours or days to get an answer.  
    To solve this problem, you can also build your own server and store transaction data in a database to build API endpoints. However, this method requires resources and maintenance, and undermines decentralization and security properties. The Graph achieved decentralization by indexing blockchain data and distributing and storing data at the same time [4]. The Graph grew more than 20 times in 20 to 21 years and handled more than 25 billion queries per month in May 2021 [2].

**2.1.2. How The Graph works**

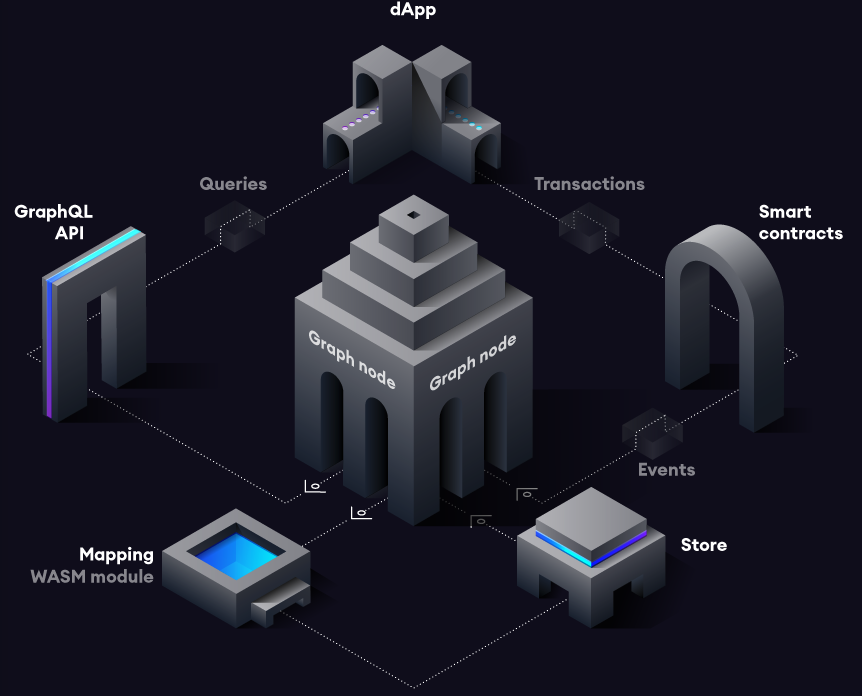


Figure 2. The data flow after deployed subgraph [4]

Figure 2 shows the flow of data after the subgraph is deployed. First, when a transaction occurs in the DApp that you want to observe, data is recorded in Ethereum. The smart contact generates one or more events while processing the transaction. At this time, Graph Node continuously observes new blocks and checks whether an Ethereum event occurs for the created subgraph. If an event corresponding to the subgraph occurs, run the mapping handler created by the user. Mapping handler generates or updates data entities stored by graph nodes by refining the generated event information through predefined code. This indexed data can be easily called from a single GraphQL endpoint from the DApp.

**2. 1. 3. Ecosystem participants**

Participants participating in the Graph Network are largely classified into four categories: delegates, indexers, curators, and consumers.

    The delegate is the owner of a GRT token that steaks to a graph project. In exchange for staking, they receive query fees that occur when consumers query data and indexer compensation that occurs when data is indexed.

    The indexer is the subject of indexing blockchain data, the main goal of The Graph. An indexer is a node operator in a Graph Network that receives compensation for data indexing, as well as query fees when a consumer requests a query to a node.

    Curator refers to the developer who creates the subgraph in The Graph. When a created subgraph is deployed and passed to the indexer, the indexer indexes it based on that subgraph. If the query created in this way is used by the consumer, the curator receives a query fee.

    Consumers are the subjects who finally pay for and inquire blockchain data indexed through The Graph. Consumers without an indexing system such as The Graph have managed their own servers and indexed data. However, with The Graph, consumers can find a subgraph that quering the desired data, request data from the graph mainnet, and pay a fee. Consumers are in a state where a large number of DApps such as Uniswap, Synthetics, Aave, and Decentraland are participating [5].

**2. 1. 4. Token Economics**

    The existing IT industry has developed technologies and tools for cloud services that can be used by billions of people. Based on the technology developed, companies typically had to reorganize their architectures and hire a large number of programmers and DevOps to manage them every time the service users reached 1,000 to 1 million or 1 billion people [6].

    Graph Network solved the above problem in a decentralized way by using an ERC-20-based token [7] called Graph Token (GRT) by software programming both incentives and penalties in the process of retrieving queries. Basically, it is a structure in which consumers pay for it and delegates, indexers, and curators earn profits. First, a curator is a developer who creates and deploys subgraphs that consumers can query. The curator receives a portion of the query fee as revenue if the created subgraph is requested by the consumer. The second is the subject that operates the graph node as an indexer. The indexer receives a subgraph created by the curator and actually indexes blockchain data and transmits query information when consumers request information. In this process, the indexer receives a query fee when a query request comes in, and receives additional incentives for indexing. Finally, the delegate simply chooses the indexer as the person who steaks the GRT to the project, delegating the GRT to a particular indexer, and similarly receives query fees and indexing incentives through that indexer [5]. What if the indexer, not the correct query delivery as above, manipulates the wrong information and provides it to the consumer? In this case, consumers or third parties can report incorrect results on the blockchain, and the blockchain itself determines whether it is true or not. If judged to be false, the indexer is disincentivized, and the incentive is paid to the person who reported this malicious behavior.

    As such, the Graph Network is built in a structure in which participants in the ecosystem linked to it can receive incentives if they pay to the network based on query consumers. It also has censorship resistance in a way that disincentives and incentives are given to misinformation.

**2. 2. Graph Network analysis**

**2. 2. 1. How is infinite expansion possible?**

    Graph Network is an incentive and disincentive system coded in the GRT token ecosystem, creating an infinitely scalable system without the involvement of third parties. The basic system is as mentioned in 2.1.4.  
    In this system, if the demand for queries by consumers who require data increases, both indexers and those who have not yet participated in the market will be able to see the increase in the size of incentives such as providing and indexing query results. Those who can secure the extra resources that have seen it work as nodes by running The Graph software to receive incentives.

In short, the more consumers' query requests increase, the more query fees will accumulate, and the better the return of those who were previously incentivized. As a result, new people who want high profits enter the market, and eventually the system is automatically adjusted to meet demand and supply without the need for server management of the central entity.

**2. 1. 2. How The Graph store datas?**

    Graph Node is a protocol for rapidly deploying DApps in Blockchain and IPFS using GraphQL [8]. In the existing industry, when the server stops working using the HTTP method of sending all the desired data from one place with the main server, all the operations of the place where the data was received from the server stopped. However, in The Graph, a random number of non-HTTP centralized IPFS servers participated in a node to transfer distributed files, and some nodes stopped working, and data was loaded from multiple nodes at once rather than from a single location, saving more than 60% of the cost for large data such as video [9]. Graph Node stores data stored through IPFS through the Postgres database.

IPFS nodes, Postgres, and Ethereum nodes are required to deploy subgraphs. After running the IPFS node and creating the postgres database and then, start the graph node with the IPFS address, Postgres database url, and Ethereum node[8].

**2. 3. Indexing data using The Graph**

**2. 3. 1. Subgraph Structure & Preparation Method**

    To create a subgraph, we need subgraph.yaml, Contract ABI, GraphQL schema and Mapping Handler [8].

    First, the subgraph.yaml is the description of the subgraph to be created, and the rest is the preparations of subgraph. In the yaml file, the GraphQL schema file created in schema can be included in the dataSources by including the address of the smart contract, ABI, and MappingHandler. Indexing is performed based on yaml.

Contract ABI is a list of functions and parameters in JSON format and must have all events, smart contracts, that you want to collect in Graph Node. For example, if you want to index Uniswap's event, Uniswap's ABI is a necessary concept.

GraphQL schema is to create a format in which a query is requested. Among the blockchain data, you can generate the value you are interested in and want to query as an entity.



Figure 3. subgraph entity example

    Figure 3 shows an example of an entity called a token. An exclamation point is used only when a field is absolutely necessary. Conversely, if the field is optional, you can write it without the exclamation mark.For Figure 3, the token must have an ID and a current owner value [8].  
    Finally, when the mapping handler is ready, the subgraph is complete. Mapping Handler takes interest event data generated by Ethereum as input values and maps it to the form of entities written in a GraphQL schema. The mapping is created in a subset language of TypeScript called AssemblyScript[10].

**2. 3. 2. Uniswap v3 subgraph deploy**

GIST SWAP is a hard copy project of Uniswap, and only the smart contract address is different, and the contents are operating with the same code. Uniswap v3 is currently being indexed and used by Total Value Locked (TVL), Token Price, etc. on the Uniswap info[11] page to The Graph's Hosted-service [12], so if you deploy the already well-established Uniswap v3 subgraph to GIST SWAP, you can only configure the same endpoint.  
Uniswap v3 subgraph is open-source [13] and attempts to clone the repository and deploy it to The Graph Hosted Service.

The deploy proceeded in the following order.

  1. git clone https://github.com/Uniswap/v3-subgraph

  2. yarn global add @graphprotocol/graph-cli

  3. yarn

  4. npm run codegen

  5. npm run build

  6. graph auth --product hosted-service <ACCESS\_TOKEN>

  7. graph deploy --product hosted-service <GIHUB USERNAME>/<SLUG>

The development dependencies are

@graphprotocol/graph-cli : ^0.20.0

@graphprotocol/graph-ts: ^0.20.0

When following the workflow in the repository, two errors occurred at the bottom.

  => The current version of graph-cli can't be used with mappings on apiVersion less than '0.0.5'

  => To use this version of the graph-cli you must upgrade the graph-ts dependency to a version greater than or equal to 0.25.0

The first error is occured from subgraph.yaml file and the error was solved by changing the apiVersion by specifying it as apiVersion of 0.0.5 or higher in the update of Graph protocol.

    Similarly, the second error was that as the Graph protocol was updated, the Assembly Script and the Graph-cli and Graph-ts libraries were updated, requiring schema migration. To solve these errors, first we change dev dependency to latest version. Command is below.

npm install --save @graphprotocol/graph-ts@latest

Then we get typescript type errors like this.



Figure 4. Type error when deploy the Uniswap v3 subgraph

In this step there are two problems. One is i32 type error.

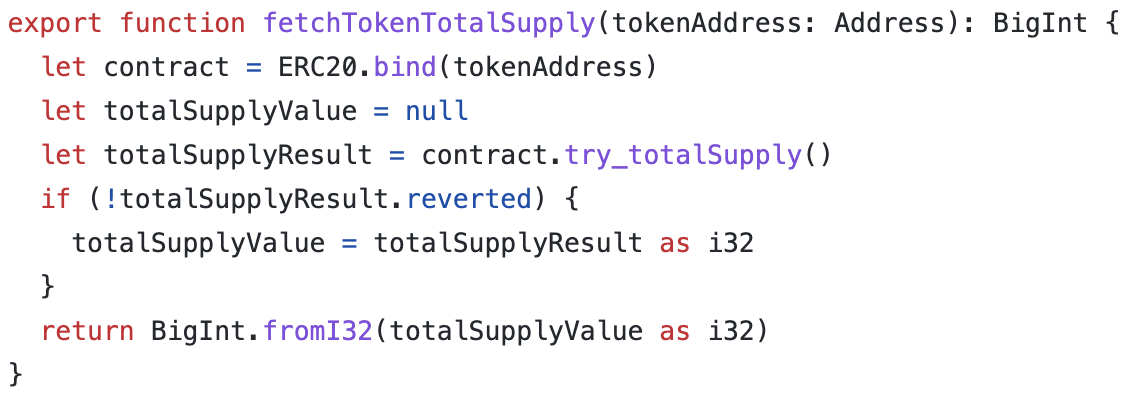


Figure 5. i32 error code

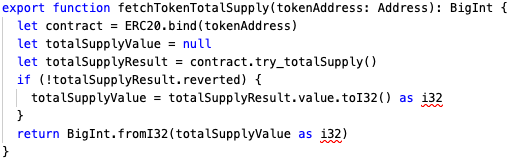


Figure 6. fix i32 error code

Because of update graph protocol. Existing Uniswap v3 subgraph code are not working. To fix this error, we should use .value.toI32() method. Then type error will fix.

And the second error is from Null type check strengthened. In the existing subgraph, there was no problem even if Null was assigned to a variable. However, as the AssemblyScript was updated, it became impossible to assign Null to a variable in subgraph[14].

Deploy was successful according to migration guide, but the sync did not proceed and an indexing error occurred. As a result, as the protocol changed, it failed to achieve its desired purpose by distributing the same Uniswap v3 subgraph as it was, and it was necessary to use a new type of subgraph.

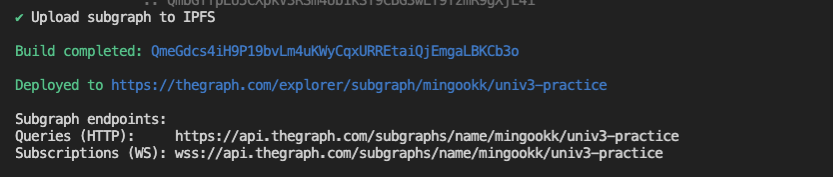


Figure 7. deploy complete on The Graph

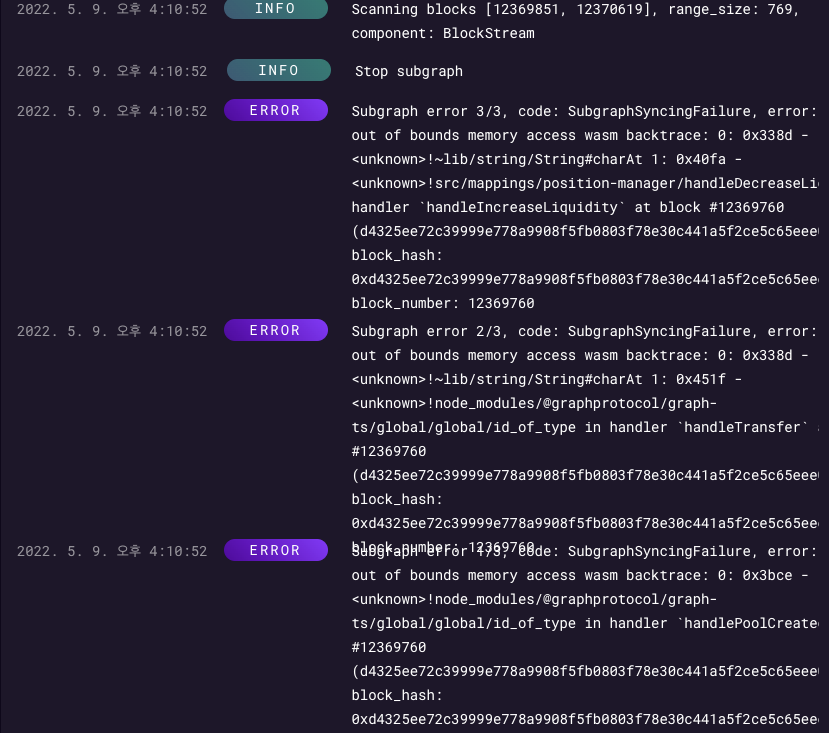


Figure 8. playground log error



Figure 9. indexing error on playground

**2. 3. 2. messari/subgraphs/uniswap-v3 deploy**

    Problems with the direct distribution of uniswap subgraphs in 2.3.1 failed to query data by direct distribution because the created subgraph was not written based on the latest protocol of The Graph. Therefore, it is necessary to have a subgraph that can retrieve Uniswap's data based on the latest protocol of The Graph.

    Messari is a company building blockchain asset data tools to provide transparency in the cryptocurrency economy [15]. The Graph, a tool for creating subgraphs, exists, but in fact, there is no guide to how to write them, so it is still difficult to create a new subgraph. Messari has a standardized subgraph schema for the Defi DApp category with the goal of "Standardized subgraphs for DeFi data" and is an open source designed to allow subgraph developers to develop subgraphs based on the same structure and based on the latest version of graph protocol. The open source was divided into categories DEX AMM, Lending Protocols, CDPs, and Yield Aggregators, and created a schema standard that can be used as a basis when creating subgraphs of DApps belonging to each category. Therefore, it is a good example for further development because the subgraphs of various Defi services developed in accordance with the form are inside the repository and the code is written based on the latest Graph Protocol.  
    Uniswap is a DApp belonging to DEX AMM, and uniswap-v3 of subgraphs is written to track Uniswap data in Optimist networks based on the corresponding standard schema [16]. The process of distributing the subgraph is as follows.

  1. git clone https://github.com/messari/subgraphs.git

  2. cd subgraphs/subgraphs/uniswap-v3

  3. yarn

  4. yarn codegen

  5. yarn build

  6. graph auth --product hosted-service <ACCESS\_TOKEN>

  7. graph deploy --product hosted-service <GIHUB USERNAME>/<SLUG>

✖ Failed to upload subgraph to IPFS: Failed to upload file to IPFS: Client network socket disconnected before secure TLS connection was established

If the following error occurs, it is a problem of the Internet environment, so if you connect to another network and try to distribute it, it will be distributed to the hosted-service normally.

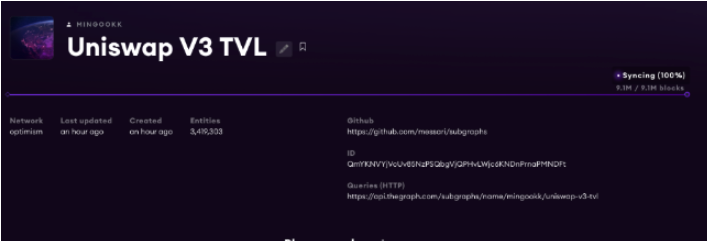


Figure 10. complete subgraph that deploy and sync data successfully

**2. 3. 2. Uniswap v3 tokens, TVL value extraction**

    The deployed subgraph has a single GraphQL endpoint in the form of HTTP, and the query request method can be requested the same as the existing GraphQL.



Figure 11. Requested query using deployed subgraphs



Figure 12. Uniswap v3 Total Value Locked value on Optimism network [17]

    To briefly explain the GraphQL query statement, you can write down the entry written in schema in advance and put the information to import the entry in parentheses. At this time, if you want to import data by sorting based on a specific field, add orderBy: <field>, orderDirection: <ascor desc>, and the desired entity data is returned in the form of a JSON object.

    Figure 11 The left side is a query statement that uses the distributed subgraph to extract five tokens in the Uniswap V3 ecosystem in ascending order by name and how much Uniswap's TVL is simultaneously, and the right side is a data object returned based on the query statement.  
    We can see that three things are called in this order of name and that the totalValueLockedUSD of Uniswap V3 on the Optimism network is retrieved almost the same as the value of the Uniswap v3 info page. Figure 11 shows how many objects are queried in the Uniswap v3 ecosystem. You can see that three tokens have been successfully loaded into the tokenens by name, and that the totalValueLockedUSD of Uniswap V3 on the Optimism network has been loaded, and Figure 12 is TVL value from Uniswap V3 info page. It displayed same data with our subgraph data.

**2. 3. 3. Future work**

    This study is a prior study to create a chart page of GIST SWAP. Since GIST SWAP does not yet have enough active pools and tokens, it chose a method of indexing Uniswap's data, which functions the same and already has enough data, and gradually implementing the functions one by one. The process to be carried out to create a chart page of GIST SWAP is as follows.

1. Uniswap v3 TVL value indexing

2. Uniswap v3 tokens indexing

3. Uniswap v3 tokens value indexing

4. Uniswap v3 Top Pools indexing

5. 1~4 code inplementing to GIST SWAP

5.1. Network change Optimism to Ropsten Network

5.2 Change smart contract address Uniswap v3 address to GIST SWAP address

In this study, we can get a TVL value, token list for Uniswap V3 (step 1~2). We can't get a token value yet from Uniswap V3. Before we implemet this codes into GIST SWAP. we should develop the subgraph about token price and pools data(3~4). After step 4, we implement this code at GIST SWAP. Now, this code tracked data on Optimism network but the GIST SWAP in on Ropsten network. So in subraph.yaml, network field are filled "optimism". It should be changed to "ropsten". And the smart contract address is also changed Uniswap's address to GIST SWAP's address. If all of this process are complete, we can make a API endpoint for making GIST SWAP chart page.

**Ⅳ. Conclusion**

    As the blockchain market grows, more and more work will be needed to query blockchain data. However, since blockchain is a database in which information of various DApps is shared in one place, querying data through a certain filter is difficult to implement and time-consuming. In this situation, we studied about The Graph Network, which has a structure that can be decentralized and expanded infinitely.

    Graph Network has created a system that distinguishes four ecosystem participants, including delegates, indexers, curators, and consumers, and allows consumers to pay query fees, and provides disincentives to nodes that report incorrect information, and continues to virtuous cycle without third party.

    We also looked at how the Graph Network could build an infinite server expansion environment and store data in a way that naturally adds providers to earn revenue if the server continues to grow and the programmer is getting a higher return on the computing resources.

    Finally, we looked at what code is needed to actually write a subgraph and how to write it, and even queried the data under various conditions of GraphQL endpoints that were actually distributed to The Graph hosted service.  
GIST SWAP is a copy project of Uniswap, and the smart contract address is different, but the contents are the same. Therefore, by changing the contract address of the currently distributed subgraph, it will be possible to try to query the data of the GIST SWAP, and develop it into a more functional subgraph by writing an additional mapping handler that can inquire the price of each token that is not available now.

**Ⅴ. Reference**

1. “DeFi Trading Hub Uniswap Surpasses $1T in Lifetime Volume”, CoinDesk https://www.coindesk.com/business/2022/05/24/defi-trading-hub-uniswap-surpasses-1t-in-lifetime-volume/
2. “Scaling Reads and Writes”, Multicoin Capital, accessed May 26, https://multicoin.capital/2021/07/30/scaling-reads-and-writes/
3. Multi-Authority Attribute-Based Access Control with Smart Contract - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Data-structure-of-an-Ethereum-transaction\_fig1\_331858454[accessed 24 May, 2022][4] “The Graph Introduction”, The Graph, https://thegraph.com/docs/en/about/introduction/
4. “The Graph Introduction”, The Graph, https://thegraph.com/docs/en/about/introduction/
5. “The Graph ecosystem-participants”, The Graph, https://thegraph.academy/ecosystem/ecosystem-participants/
6. “Infinite Scale”, Multicoin Capital, accessed May 26, https://multicoin.capital/2020/02/06/infinite-scale/
7. “Graph Token”, etherscan, accessed May 26, https://etherscan.io/token/0xc944e90c64b2c07662a292be6244bdf05cda44a7#tokenInfo
8. The Graph (2022) graph-node [source code], https://github.com/graphprotocol/graph-node/blob/master/docs/getting-started.md
9. ipfs, accessed May 26 https://ipfs.io
10. “create-subgraph-hosted”, The Graph, accessed May 26, https://thegraph.com/docs/en/developer/create-subgraph-hosted/
11. “Uniswap info”, Uniswap v3, accessed May 26 https://info.uniswap.org/
12. “Uniswap V3 Subgraph”, The Graph. accessed May 26, https://thegraph.com/hosted-service/subgraph/ianlapham/uniswap-v3-subgraph
13. Uniswap (2022) Uniswap v3 subgraph [source code], https://github.com/Uniswap/v3-subgraph
14. “Assemblyscript migration guide”, The Graph, accessed May 26, https://thegraph.com/docs/developer/assemblyscript-migration-guide
15. "Messari about", Messari, accessed May 28, https://messari.io/about
16. messari (2022) subgraphs [source code], https://github.com/messari/subgraphs
17. "Uniswap Info optimism", Uniswap info, accessed May 28, https://info.uniswap.org/#/optimism/