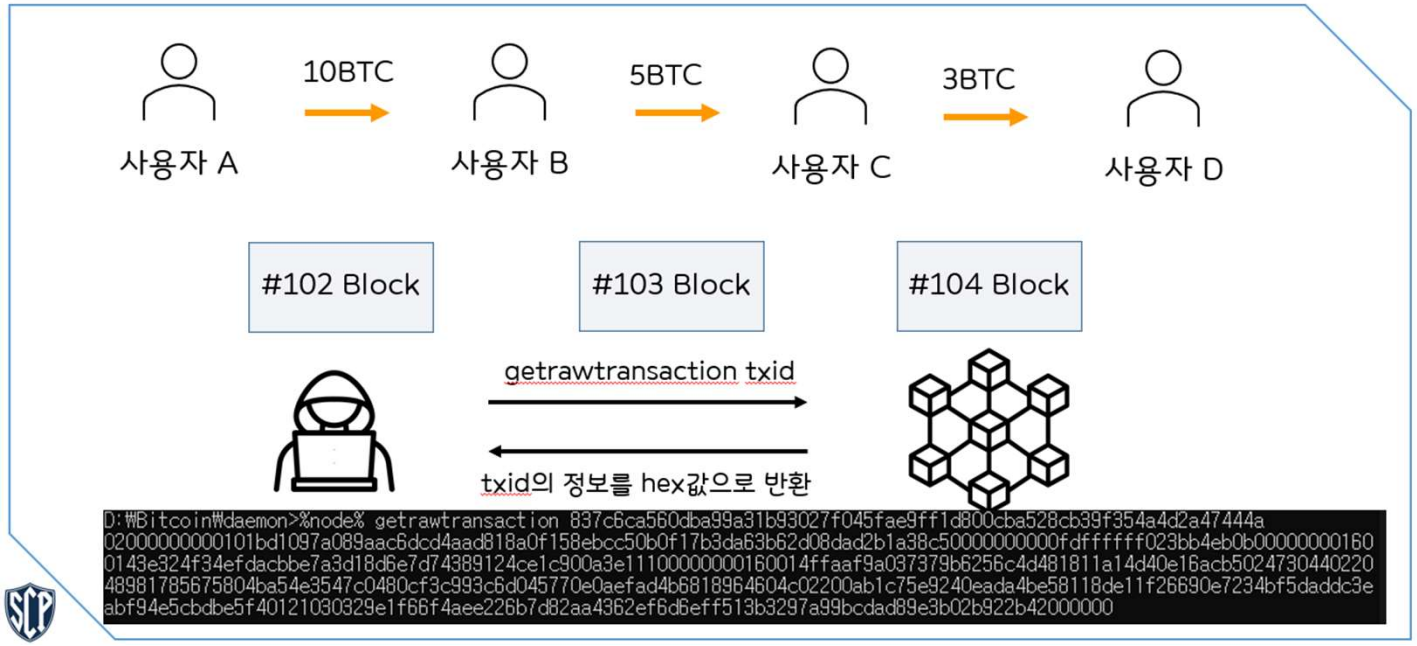
블록체인 자산추적(2)

91714167 유재겸

목차

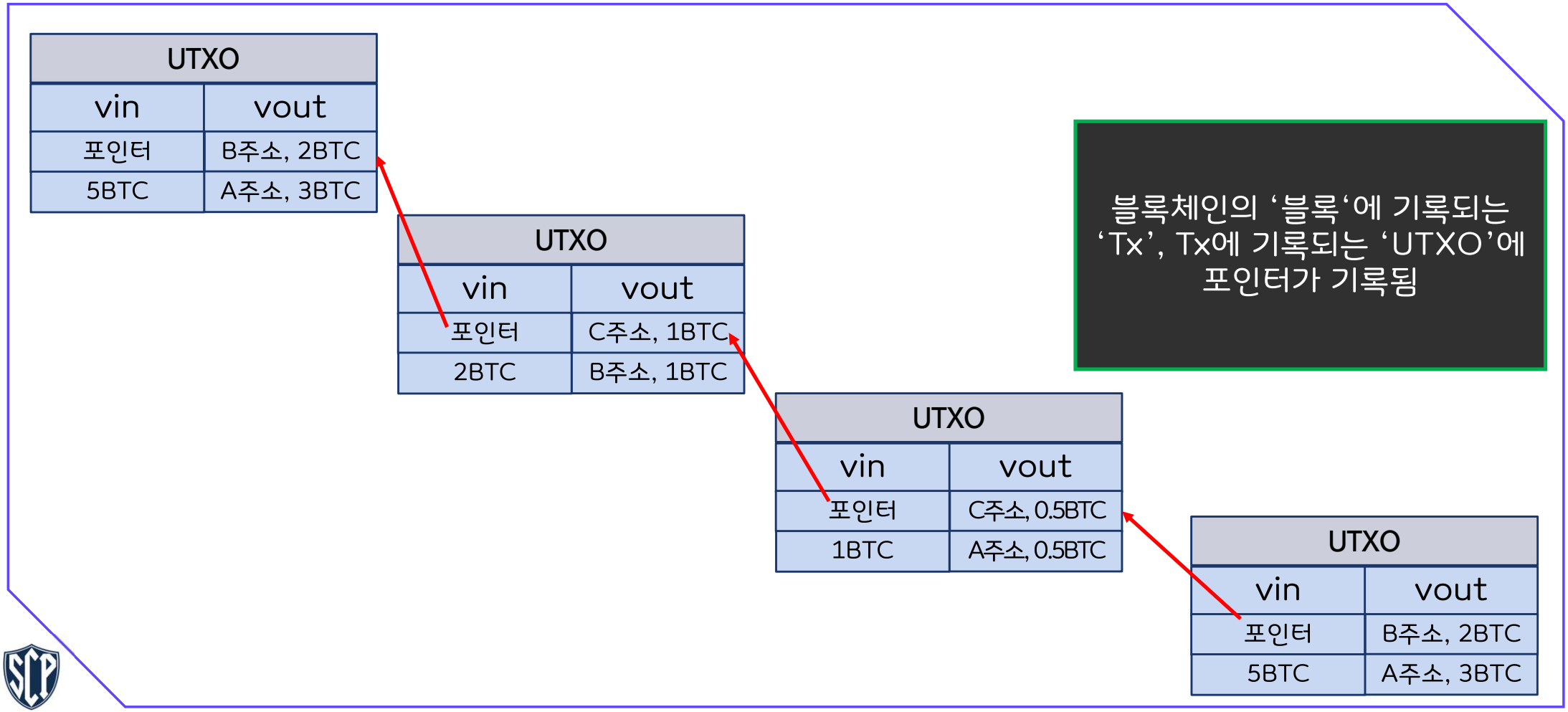
01 지난 이야기

지난 이야기

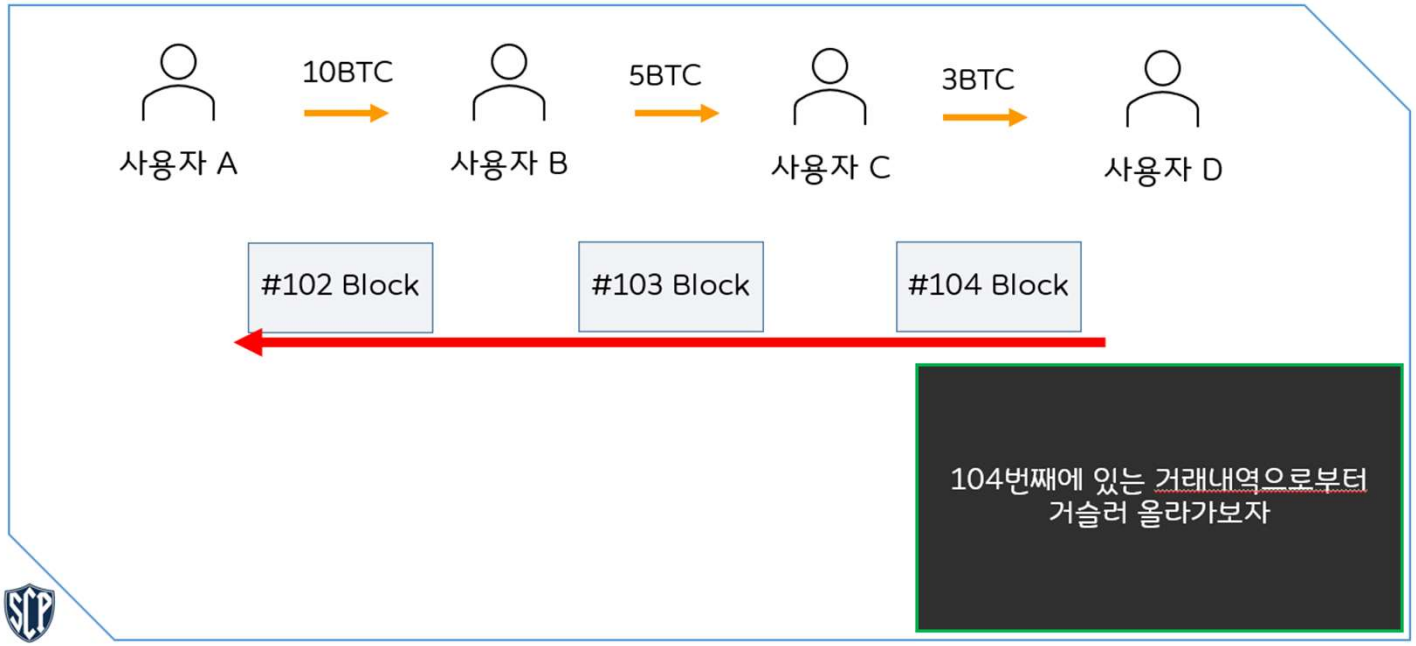


지난 발표에서 블록체인
자산추적에 관한 이야기와 실습

지난 이야기

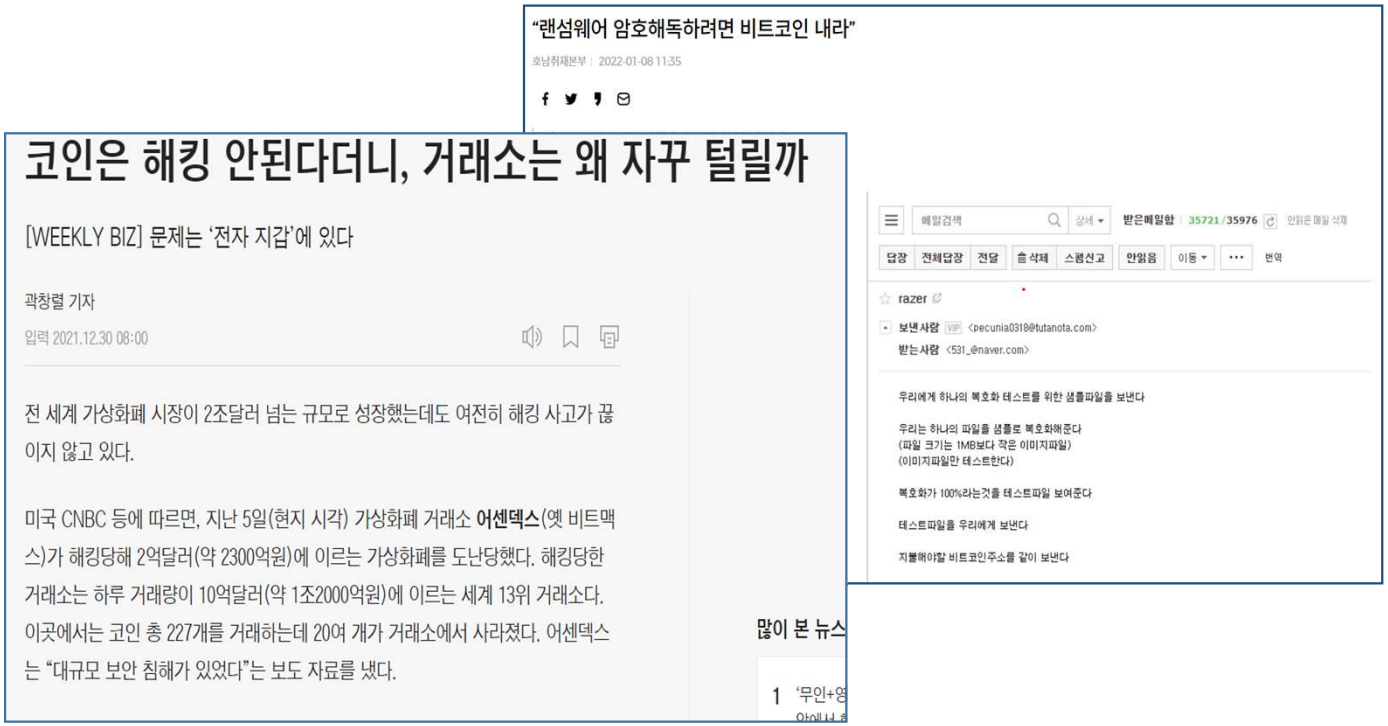

UTXO
vin
vout
포인터
B주소, 2BTC
5BTC
A주소, 3BTC
UTXO
vin
vout
포인터
C주소, 1BTC
2BTC
B주소, 1BTC
UTXO
vin
vout
포인터
C주소, 0.5BTC
1BTC
A주소, 0.5BTC
UTXO
vin
vout
포인터
B주소, 2BTC
5BTC
A주소, 3BTC
블록체인의 '블록'에 기록되는 'Tx', Tx에 기록되는 'UTXO'에 포인터가 기록됨

지난 이야기



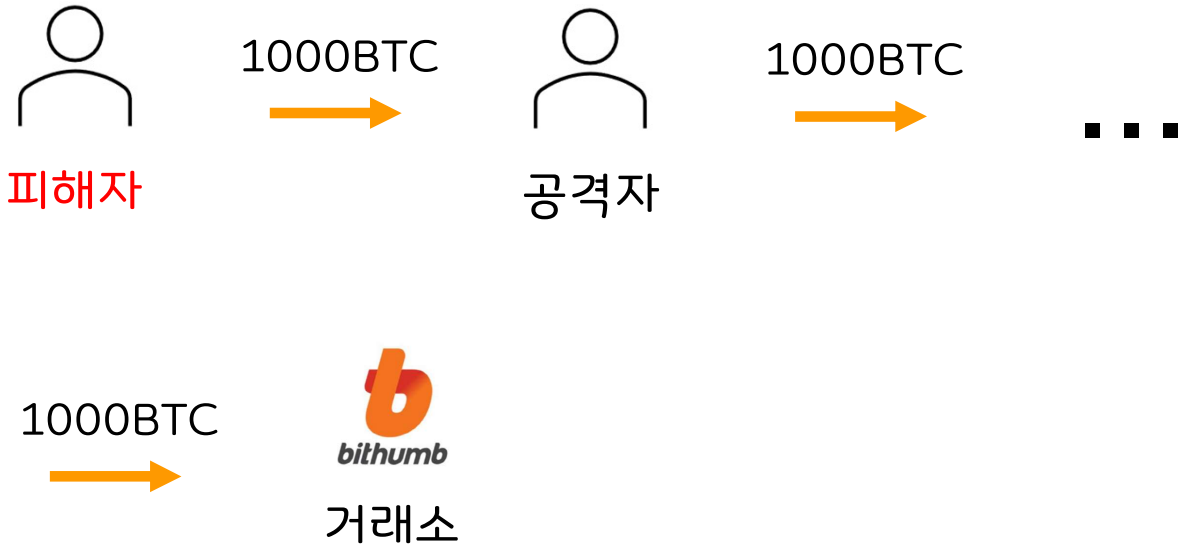
이를 통해 거래를 역추적,
그에 대한 실습을 진행

지난 이야기



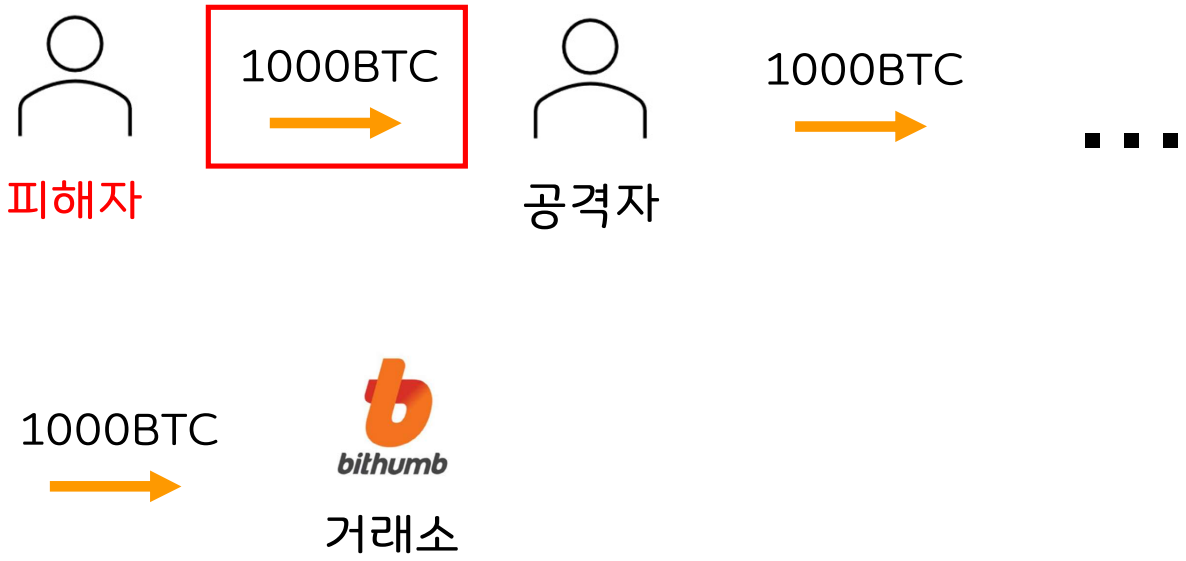
자산추적의 목표는,
불법거래나 범죄에 악용되는
가상화폐를 추적하기 위함

지난 이야기

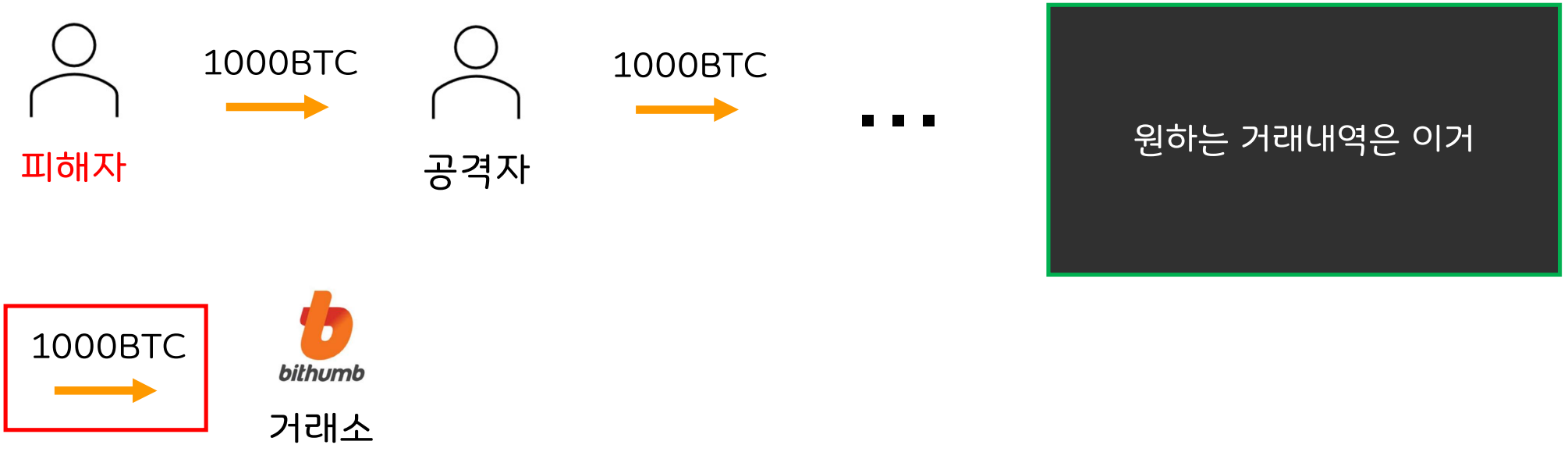


거래소가 털린 상황을 가정

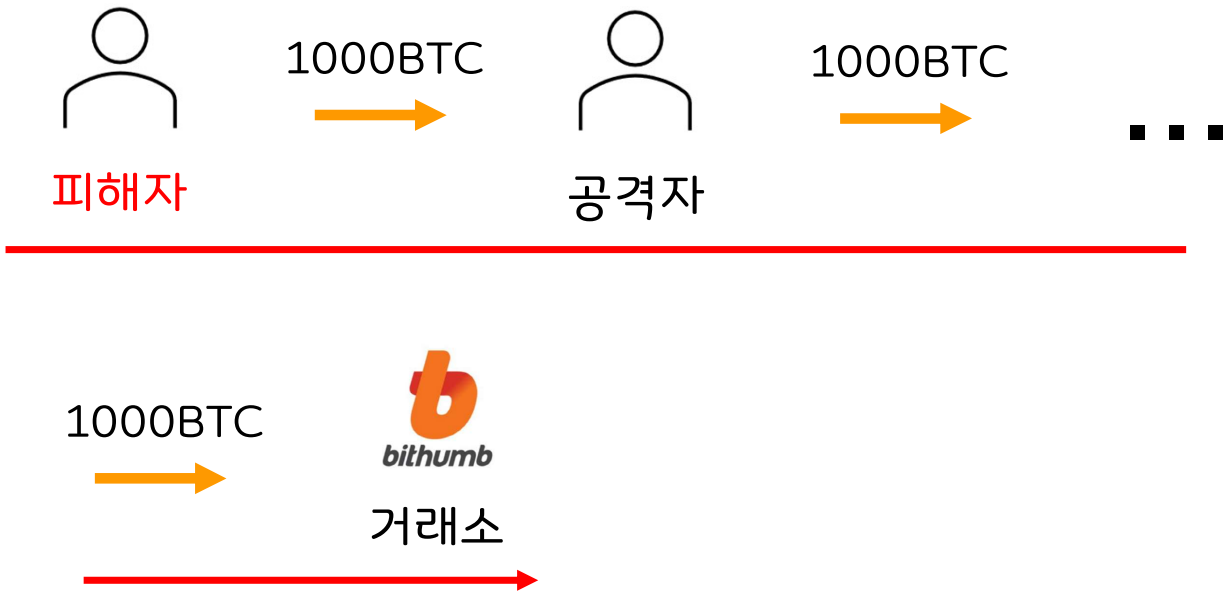
지난 이야기


1000BTC
피해자
1000BTC
공격자
1000BTC
. . .
해당 거래내역에 대해서는
알 수 있음
1000BTC
bithumb
거래소

지난 이야기

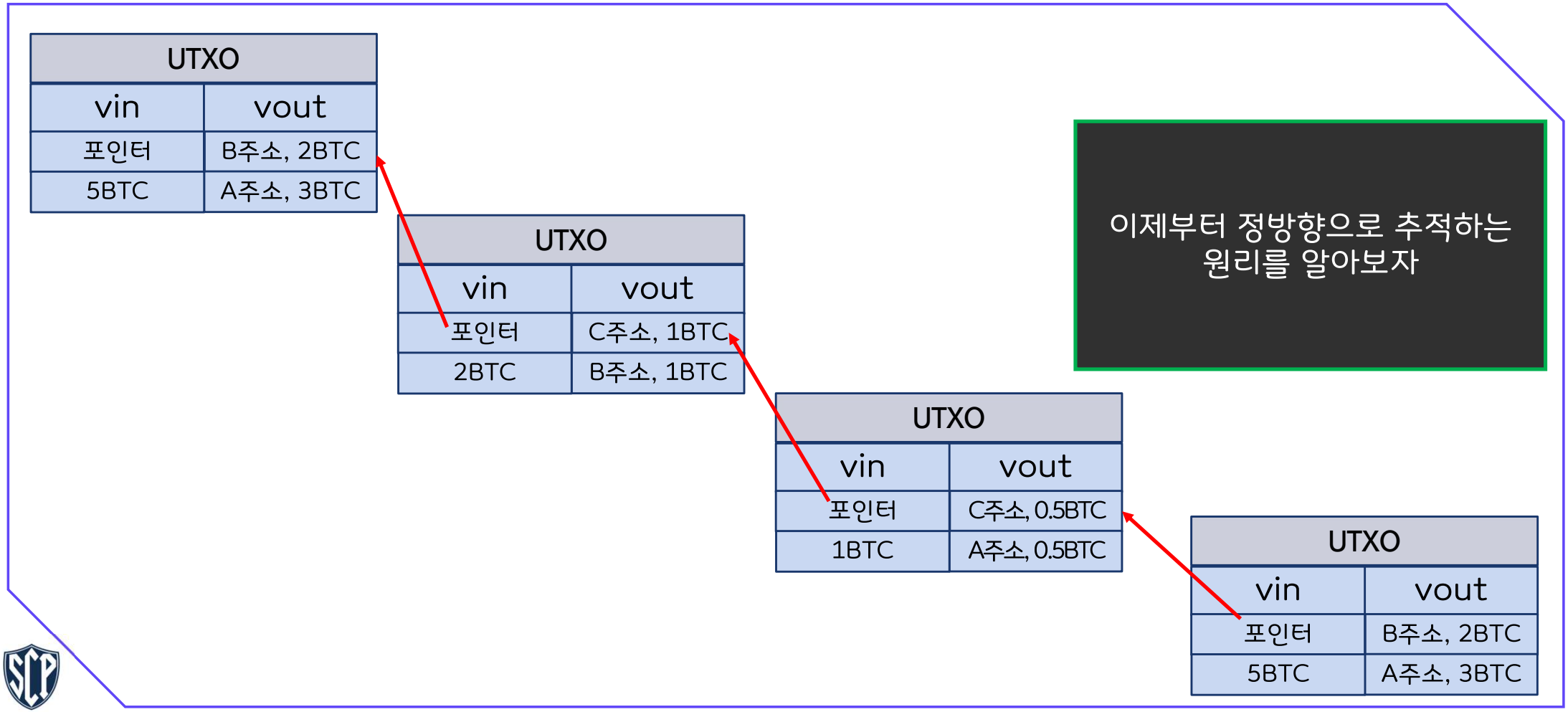


지난 이야기



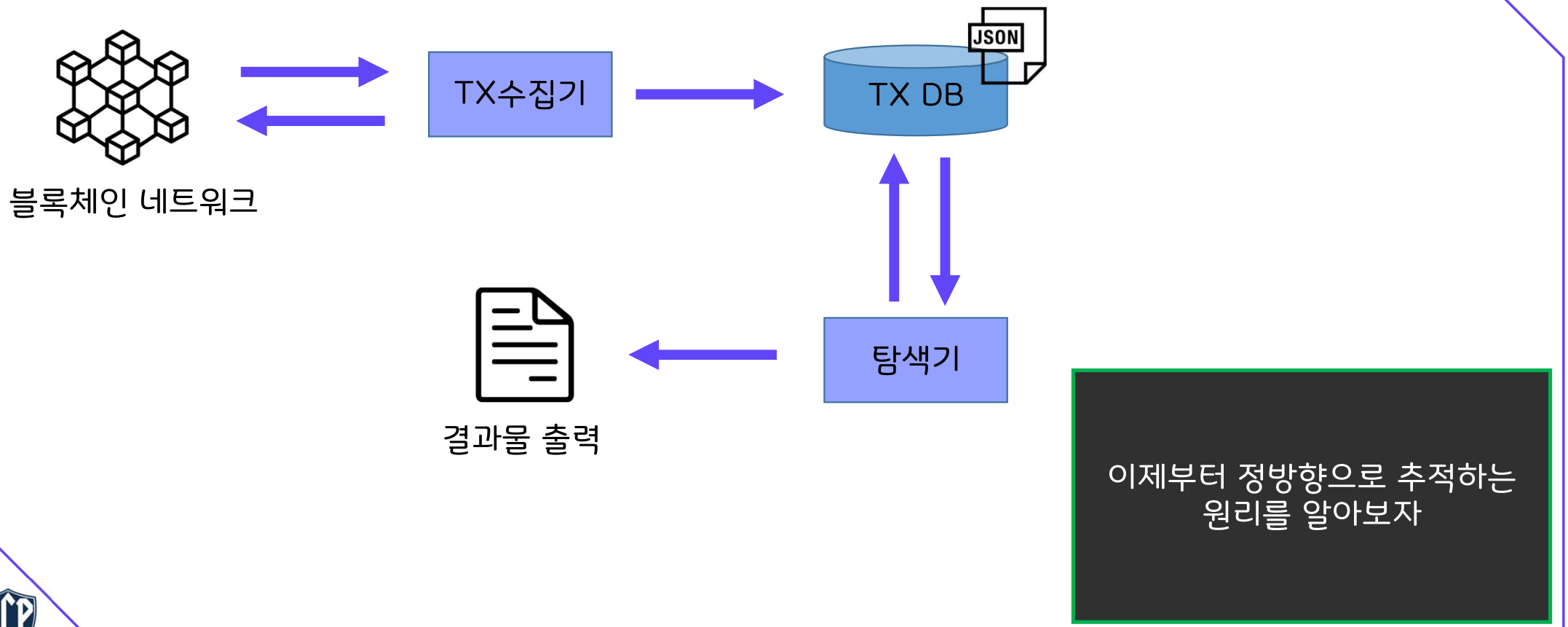
즉 거래추적은 역방향이 아닌,
정방향이 되어야한다

지난 이야기

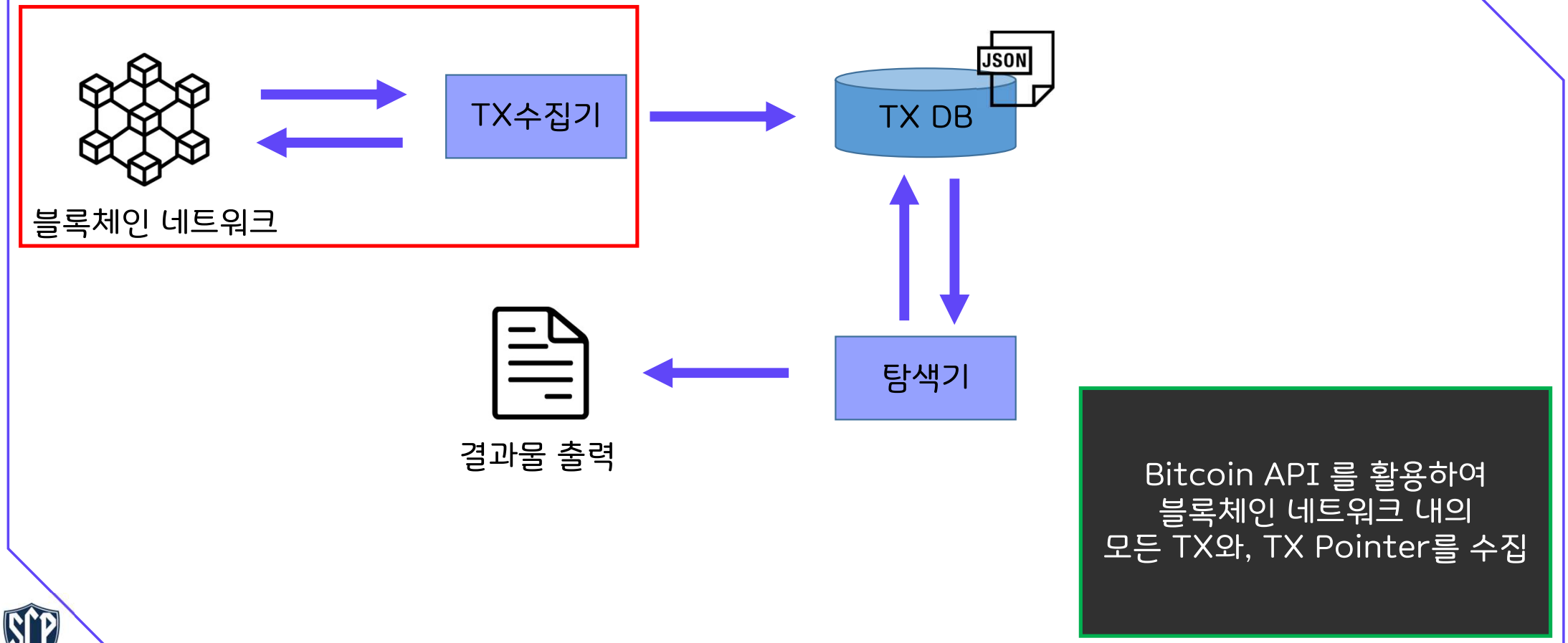

UTXO
vin
vout
포인터
B주소, 2BTC
5BTC
A주소, 3BTC
UTXO
vin
vout
포인터
C주소, 1BTC
2BTC
B주소, 1BTC
UTXO
vin
vout
포인터
C주소, 0.5BTC
1BTC
A주소, 0.5BTC
UTXO
vin
vout
포인터
B주소, 2BTC
5BTC
A주소, 3BTC
이제부터 정방향으로 추적하는
원리를 알아보자

02 원리 및 구조도

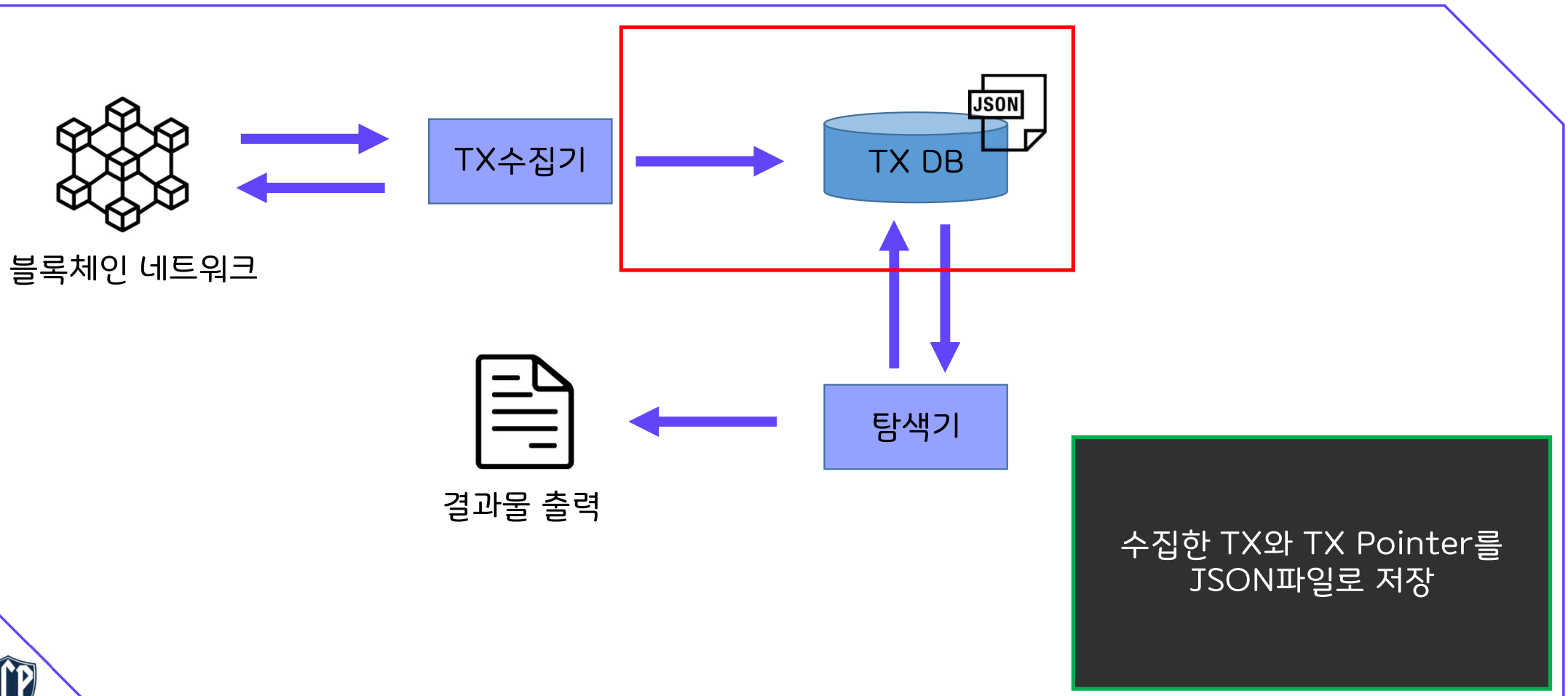
원리 및 구조도



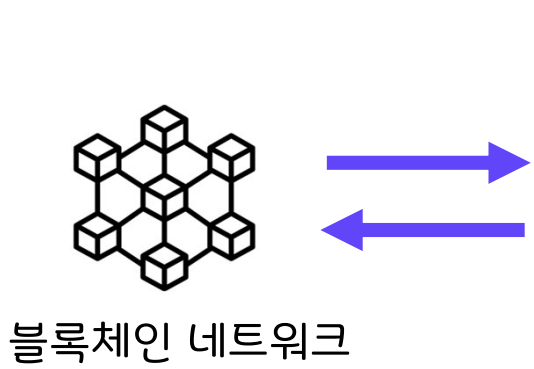
원리 및 구조도


TX수집기
JSON
TX DB
블록체인 네트워크
탐색기
결과물 출력
Bitcoin API 를 활용하여
블록체인 네트워크 내의
모든 TX와, TX Pointer를 수집

원리 및 구조도


블록체인 네트워크
TX수집기
JSON
TX DB
탐색기
결과물 출력
수집한 TX와 TX Pointer를
JSON파일로 저장

원리 및 구조도



블록체인 네트워크

TX수집기

JSON

TX DB

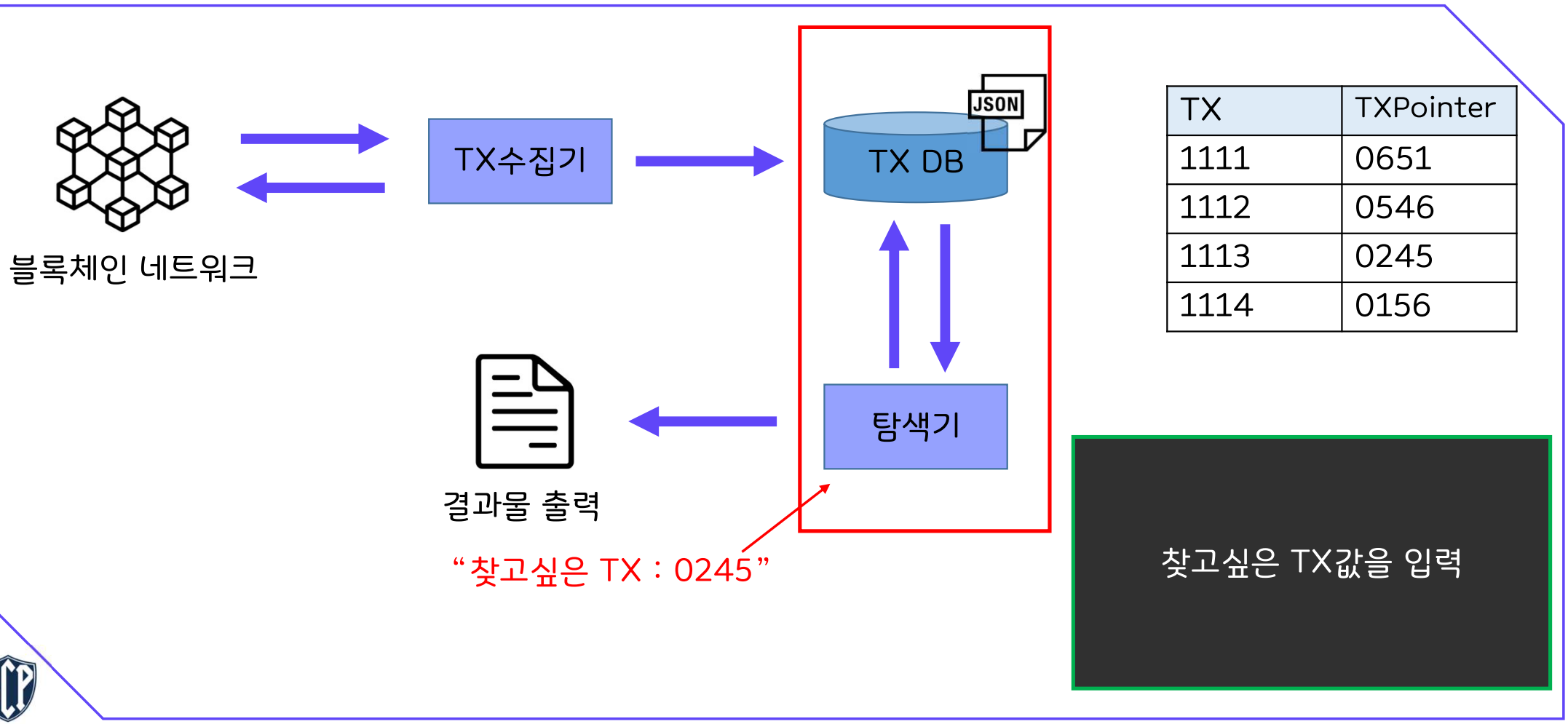
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

탐색기

결과물 출력

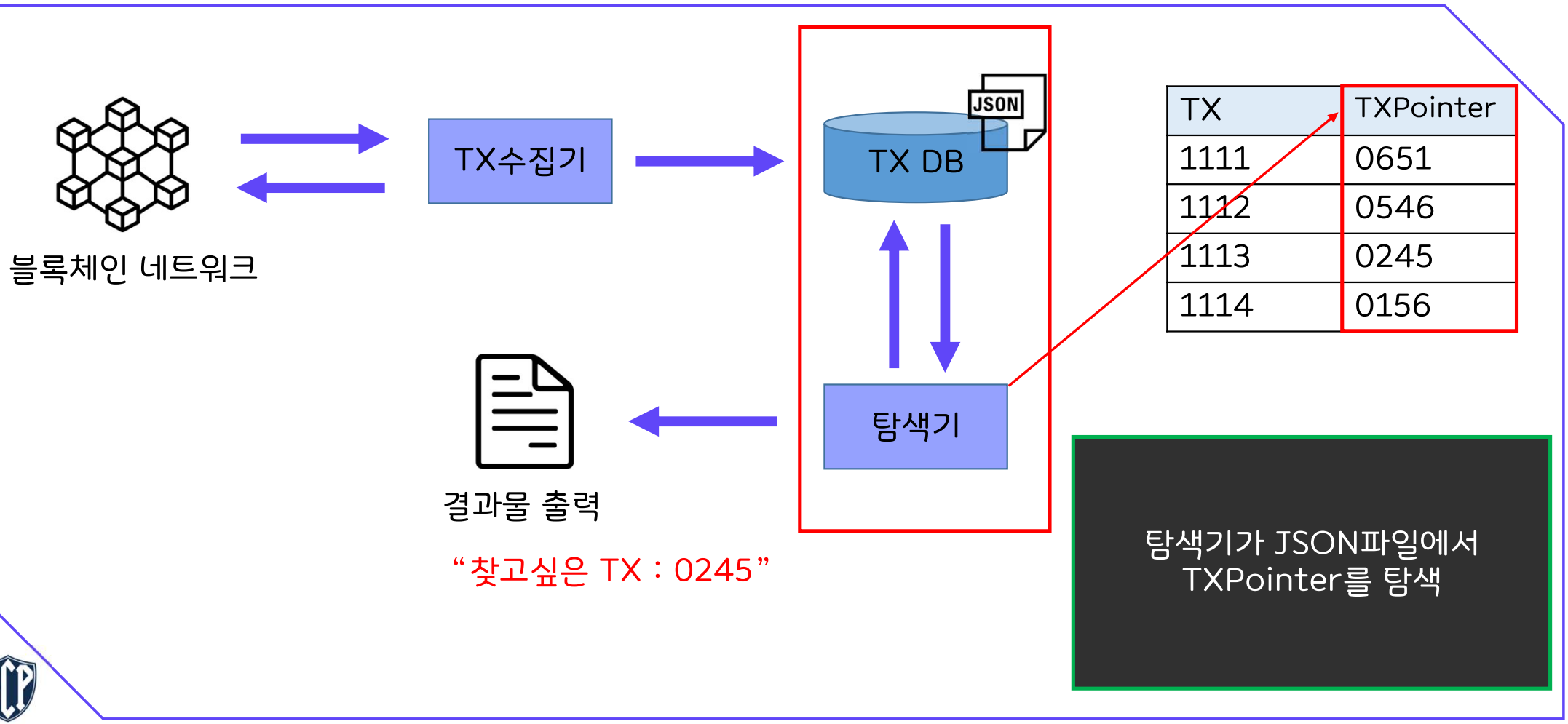
수집한 TX와 TX Pointer를
블록별로 JSON파일로 저장

원리 및 구조도

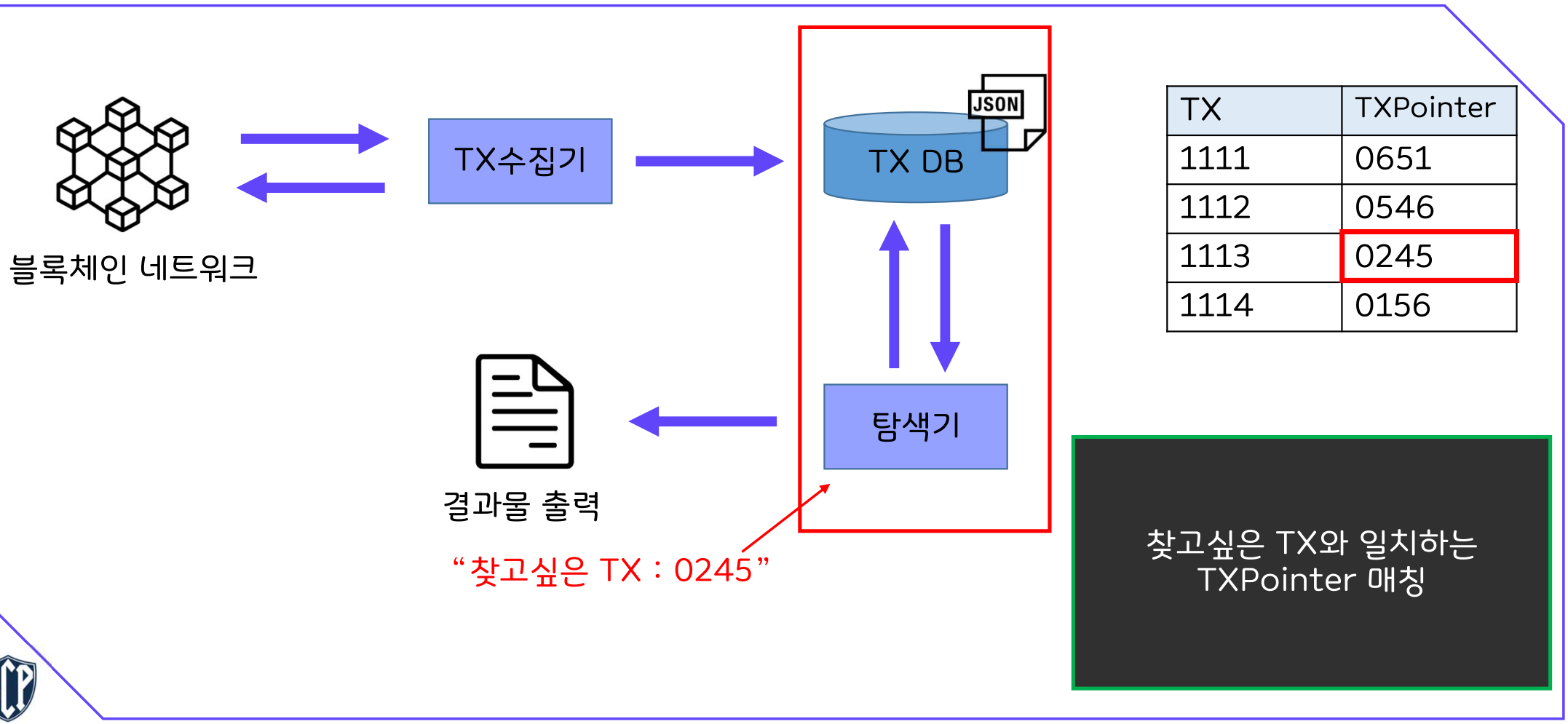


TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

원리 및 구조도


블록체인 네트워크
TX수집기
TX DB
JSON
탐색기
결과물 출력
"찾고싶은 TX : 0245"
TX
TXPointer
1111
0651
1112
0546
1113
0245
1114
0156
탐색기가 JSON파일에서
TXPointer를 탐색

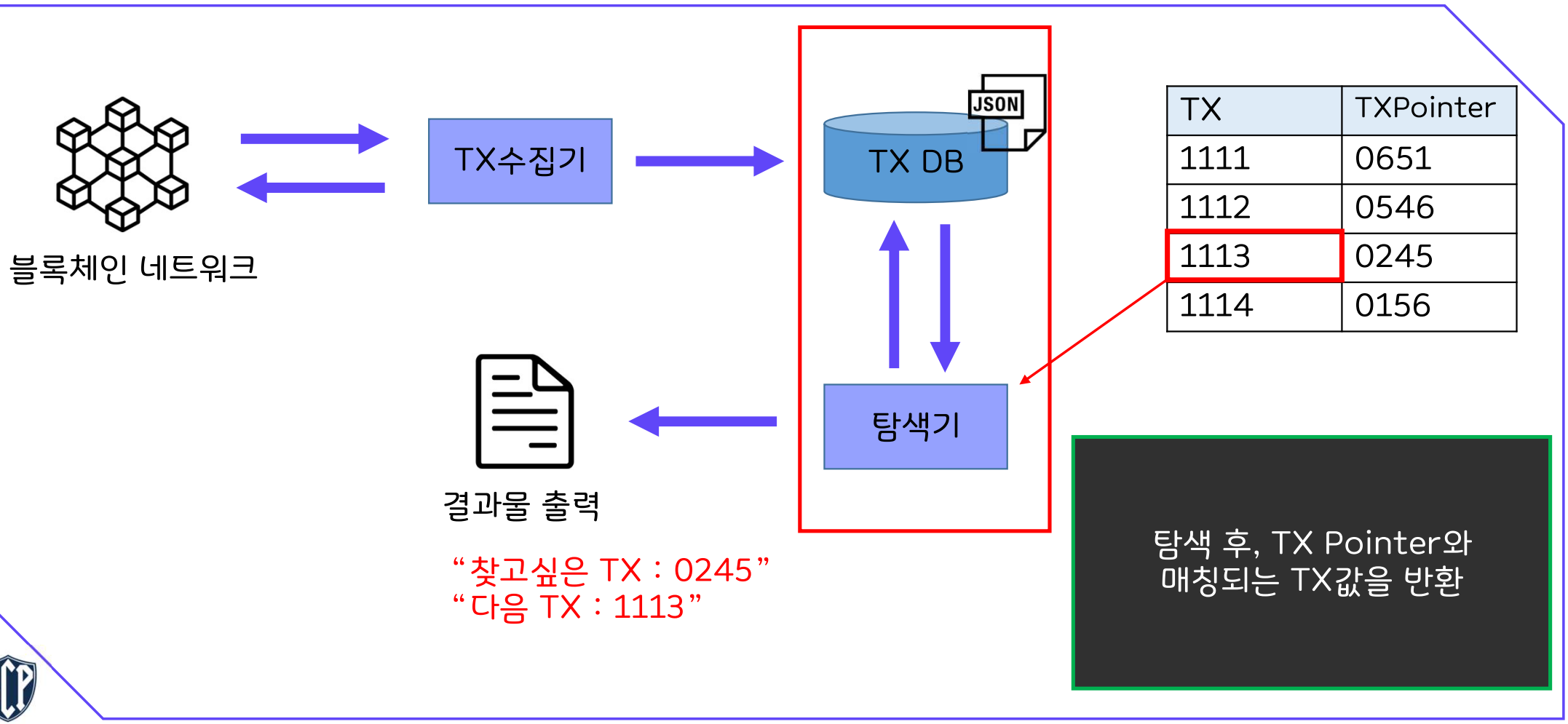
원리 및 구조도



TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

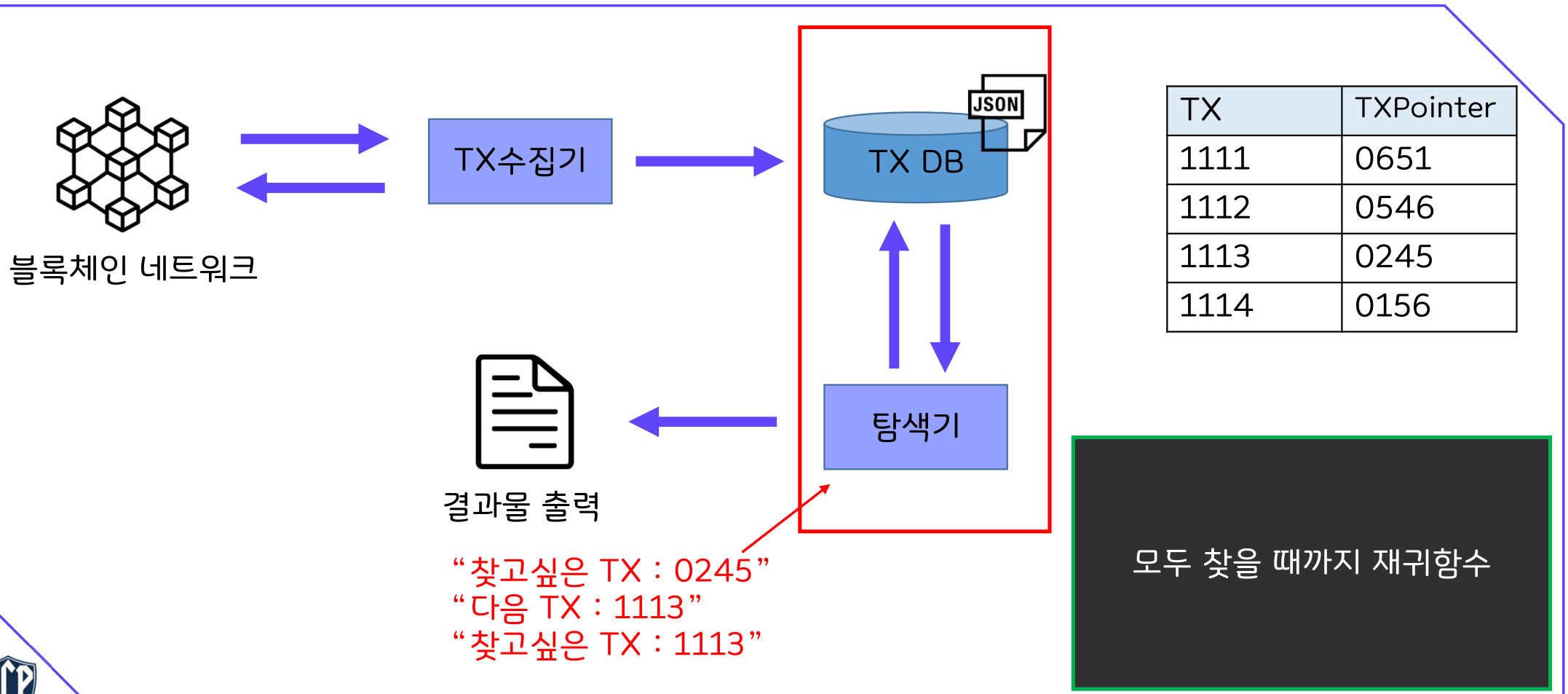
찾고싶은 TX와 일치하는
TXPointer 매칭

원리 및 구조도



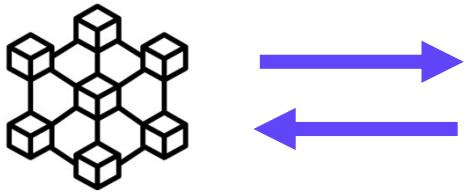
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

원리 및 구조도



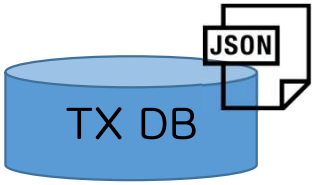
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

원리 및 구조도



블록체인 네트워크

TX수집기



TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

“찾고싶은 TX : 0245”
“다음 TX : 1113”
“찾고싶은 TX : 1113”
“다음 TX : 1231”
.
.
.

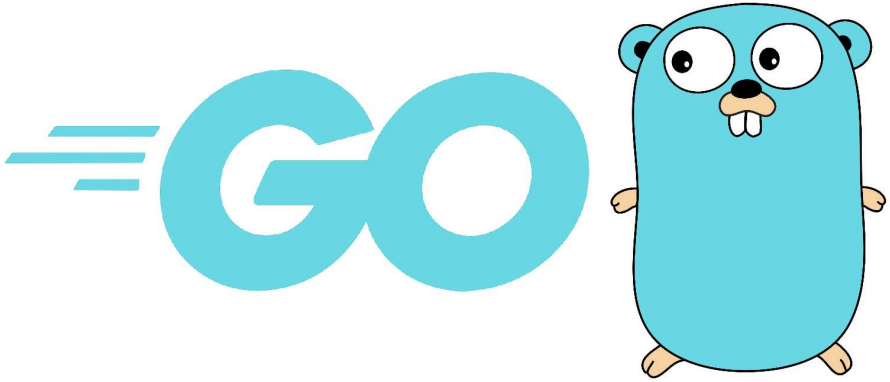
결과물 출력

탐색기

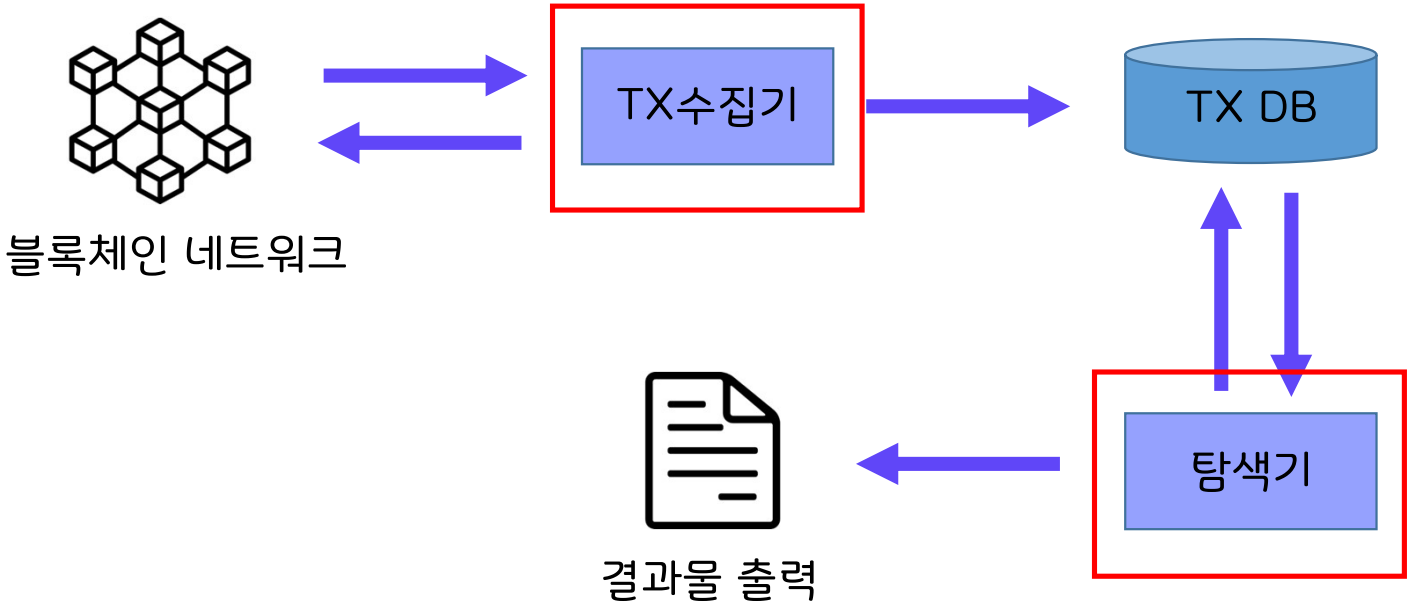
최종 결과물

03 개발 및 구현

개발 및 구현

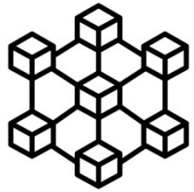


개발 및 구현

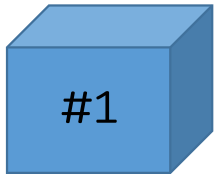


개발 및 구현

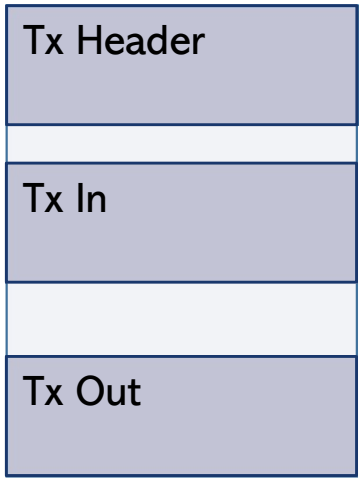
TX수집기



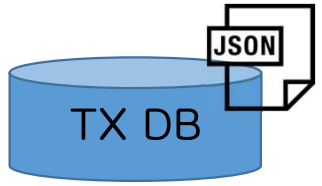
블록체인 네트워크



Block



TX



TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

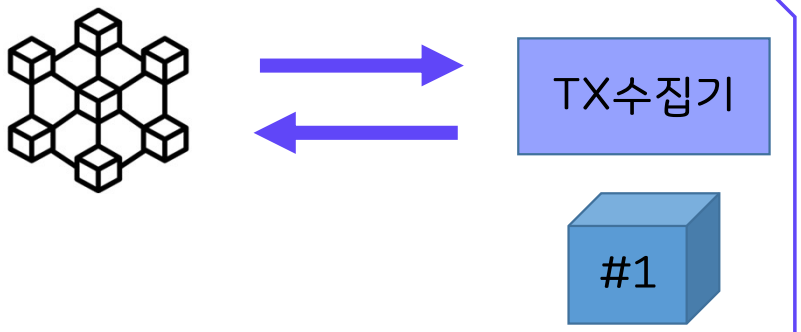
개발 및 구현

```go
func loadblock(blocknumber string) []string {
    var blockhash []byte

    blockhash = nodecmd("getblockhash", blocknumber)
    //원하는 블록의 해시값을 가져옴

    var blockinfo []byte
    blockinfo = nodecmd("getblock", string(blockhash))
    //해당 블록의 정보를 불러옴

    matchtxs, _ := regexp.Compile("\\[[^]]*\\]")
    txs := matchtxs.FindString(string(blockinfo))
    matchtx, _ := regexp.Compile("[^\"]*")
    txlist := matchtx.FindAllString(txs, -1)
    //정규표현식 파싱을 통하여 블록의 트랜잭션 값만 불러옴
    txlist = txlist[1:]
    //파싱을 통한 값 첫번째에 공백이 들어가 있어서 해당부분 제거
    return txlist
}
```



비트코인 api를 활용하여,
블록을 상세정보 요청

개발 및 구현

```
D:₩Bitcoin₩daemon>%node% getblock 14d049b33a10a23abadd6e3cd66490ae7635616e151ffbcaa14aa9867a94d663
{
  "hash": "14d049b33a10a23abadd6e3cd66490ae7635616e151ffbcaa14aa9867a94d663",
  "confirmations": 2,
  "height": 103,
  "version": 536870912,
  "versionHex": "20000000",
  "merkleroot": "a8761b40cd8c90be82102c6ed9b33662376181cfb80ca916435aa948388bdd33",
  "time": 1658507291,
  "mediantime": 1658410247,
  "nonce": 1,
  "bits": "207fffff",
  "difficulty": 4.656542373906925e-10,
  "chainwork": "00000000000000000000000000000000000000000000000000000000000000d0",
  "nTx": 7,
  "previousblockhash": "1d8bb0e204d32c547e178a8a47b75cc1121f921b41c161896dbb8ef287c63b08",
  "nextblockhash": "7975cd943b9495f893ffe41603b1feccdd355a588a564aa7b0c42d15749be40a",
  "strippedsize": 933,
  "size": 1729,
  "weight": 4528,
  "tx": [
    "27f8ae4a5b812f765ae38923e4c4ebe83fc6ad5ec464e6a053928fead3f24e94",
    "4fa0beaa41cc7c180215c5ce19ed59bb7c90b7adde15acec3d76a9e7388aab15",
    "1059678796083d29953aa7771d70a4e36dadd5d698de080fcea8ef8d84fcb15a",
    "9dca1dfcd68b2536747776de7ae38b13873f18605a5bad42cd54ec2f4adfde5f",
    "effdab52c683c703af3848386f47c4e185dc1de67031a91dab4abe753f623c68",
    "46b9f166584cef2e8d6e71274831a2206c89ed852b60f7149799a8f2d4100071",
    "742718dc9625c6aac6eca4b57d93051208540cd3d65773a41131ce77eee16291"
  ]
}
```

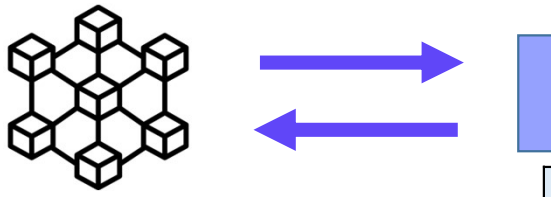
개발 및 구현

```go
func loadblock(blocknumber string) []string {
    var blockhash []byte

    blockhash = nodecmd("getblockhash", blocknumber)
    //원하는 블록의 해시값을 가져옴

    var blockinfo []byte
    blockinfo = nodecmd("getblock", string(blockhash))
    //해당 블록의 정보를 불러옴

    matchtxs, _ := regexp.Compile("\\[[^]]*\\]")
    txs := matchtxs.FindString(string(blockinfo))
    matchtx, _ := regexp.Compile("[^\"]*")
    txlist := matchtx.FindAllString(txs, -1)
    //정규표현식 파싱을 통하여 블록의 트랜잭션 값만 불러옴
    txlist = txlist[1:]
    //파싱을 통한 값 첫번째에 공백이 들어가 있어서 해당부분 제거
    return txlist
}
```



TX
1111
1112
1113
1114

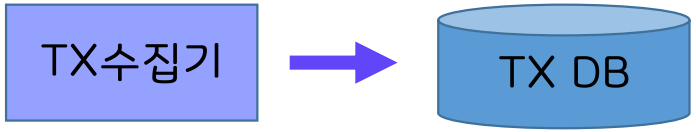
블록안에는 다양한 정보들이 있기 때문에 파싱을 통하여 TX목록만 얻어낸다

개발 및 구현

```
func savetx(txlist []string, blocknumber string) {

    data := make([]TxData, (len(txlist) / 2))
    for i := 0; i < (len(txlist) / 2); i++ {
        getrawtx := string(nodecmd("getrawtransaction", txlist[2*i]))
        //getrawtransaction 으로 해당 트랜잭션의 정보를 hex값으로 얻어옴
        txinfo := string(nodecmd("decoderawtransaction", getrawtx))
        //얻어온 hex값 해독
        vinmatch, _ := regexp.Compile("\\[[^]]*\\]")
        vin := vinmatch.FindString(txinfo)
        //vin 파싱
        pointermatch, _ := regexp.Compile("[a-z0-9]{63,64}")
        pointer := pointermatch.FindAllString(vin, -1)
        data[i].Info = txlist[2*i]
        data[i].Pointer = pointer
    }
    doc, _ := json.Marshal(data) // data를 JSON 문서로 변환

    err2 := ioutil.WriteFile("../txinfo/"+blocknumber+".json", doc, os
    if err2 != nil {
        fmt.Println(err2)
    }
```



TX
1111
1112
1113
1114

TXPointer가 필요,
TX상세정보 요청

개발 및 구현

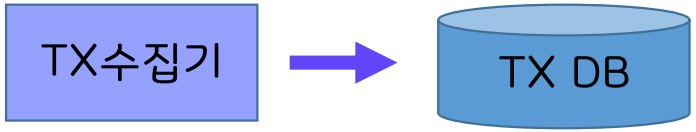
```
D:₩Bitcoin₩daemon>%node% decoderawtransaction 020000000001011a6b41dfa94e3dff6c19a1c292e787e479a9d411485daf25555761790e2fb56
6f91504ae6f2a726500ca9a3b00000000160014f5aabb91c690510d273f1e42988946ccf5c158d80247304402201ead9cecf892774248af1cc5ef90fe81
c737b2519981683001b7073253e1dc99d50121020a61780e532d2faefb40829a2ea95fe66d44352965c603b30ec201e4b44c888666000000
{
  "txid": "4fa0beaa41cc7c180215c5ce19ed59bb7c90b7adde15acec3d76a9e7388aab15",
  "hash": "087176c15bf71a48b916b545febeaa81c602233d7e4959ecdfe9677dcf2700db",
  "version": 2,
  "size": 222,
  "vsize": 141,
  "weight": 561,
  "locktime": 102,
  "vin": [
    {
      "txid": "61b52f0e7961575525af5d4811d4a979e487e792c2a1196cff3d4ea9df416b1a",
      "vout": 0,
      "scriptSig": {
        "asm": "",
        "hex": ""
      },
      "txinwitness": [
        "304402201ead9cecf892774248af1cc5ef90fe81b8a6b50b6b98bb26747e70650655465e02200d9e7e7555bfe56f74787ed8cad6edc737b251
        "020a61780e532d2faefb40829a2ea95fe66d44352965c603b30ec201e4b44c8886"
      ],
      "sequence": 4294967293
    }
  ],
  "vout": [
    {
      "value": 39.99996475,
      "n": 0,
      "scriptPubKey": {
        "asm": "0 2b04ebf9ae8b05895d5c0b86f91504ae6f2a7265",
        "desc": "addr(bcrt1q9vzwh7dw3vzcjh2upwr0j9gy4ehj5un9jk8pzj)#7kn2svfs",
        "hex": "00142b04ebf9ae8b05895d5c0b86f91504ae6f2a7265",
        "address": "bcrt1q9vzwh7dw3vzcjh2upwr0j9gy4ehj5un9jk8pzj",
        "type": "witness_v0_keyhash"
      }
```

개발 및 구현

```go
func savetx(txlist []string, blocknumber string) {

    data := make([]TxData, (len(txlist) / 2))
    for i := 0; i < (len(txlist) / 2); i++ {
        getrawtx := string(nodecmd("getrawtransaction", txlist[2*i]))
        //getrawtransaction 으로 해당 트랜잭션의 정보를 hex값으로 얻어옴
        txinfo := string(nodecmd("decoderawtransaction", getrawtx))
        //얻어온 hex값 해독
        vinmatch, _ := regexp.Compile("\\[[^]]*\\]")
        vin := vinmatch.FindString(txinfo)
        //vin 파싱
        pointermatch, _ := regexp.Compile("[a-z0-9]{63,64}")
        pointer := pointermatch.FindAllString(vin, -1)
        data[i].Info = txlist[2*i]
        data[i].Pointer = pointer
    }
    doc, _ := json.Marshal(data) // data를 JSON 문서로 변환

    err2 := ioutil.WriteFile("../txinfo/"+blocknumber+".json", doc, os
    if err2 != nil {
        fmt.Println(err2)
    }
```



TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

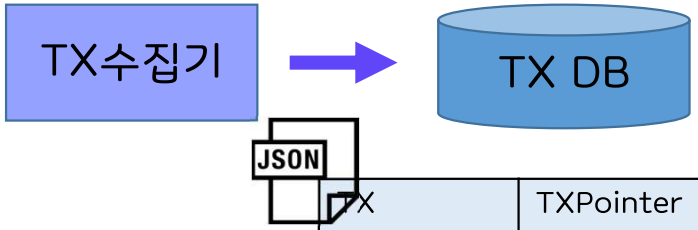
트랜잭션 안에는 다양한 정보가 있기 때문에,
파싱하여 Pointer값만 저장

개발 및 구현

```go
func savetx(txlist []string, blocknumber string) {

    data := make([]TxData, (len(txlist) / 2))
    for i := 0; i < (len(txlist) / 2); i++ {
        getrawtx := string(nodecmd("getrawtransaction", txlist[2*i]))
        //getrawtransaction 으로 해당 트랜잭션의 정보를 hex값으로 얻어옴
        txinfo := string(nodecmd("decoderawtransaction", getrawtx))
        //얻어온 hex값 해독
        vinmatch, _ := regexp.Compile("\\[[^]]*\\]")
        vin := vinmatch.FindString(txinfo)
        //vin 파싱
        pointermatch, _ := regexp.Compile("[a-z0-9]{63,64}")
        pointer := pointermatch.FindAllString(vin, -1)
        data[i].Info = txlist[2*i]
        data[i].Pointer = pointer
    }
    doc, _ := json.Marshal(data) // data를 JSON 문서로 변환

    err2 := ioutil.WriteFile("../txinfo/"+blocknumber+".json", doc, os
    if err2 != nil {
        fmt.Println(err2)
    }
```



JSON

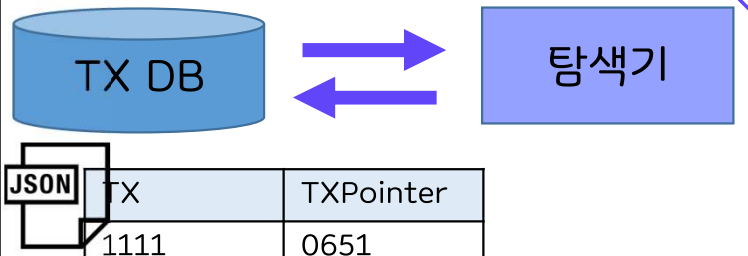
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

마지막으로 TX와 TXPointer를
JSON형식으로 변환하여
TXDB에 저장

개발 및 구현

```go
func searchnexttx(searchtxinfo string, blocknumber int) {
    var list []string
    for s := 1; s < blocknumber; s++ {
        stostring := strconv.Itoa(s)
        b, err := ioutil.ReadFile("../txinfo/" + stostring + ".json")
        if err != nil {
            fmt.Println(err)
        }
        var data []TxData
        json.Unmarshal(b, &data)
        for t := 0; t < len(data); t++ {
            for u := 0; u < len(data[t].Pointer); u++ {
                if data[t].Pointer[u] == searchtxinfo {
                    list = append(list, data[t].Info)
                }
            }

        }
```



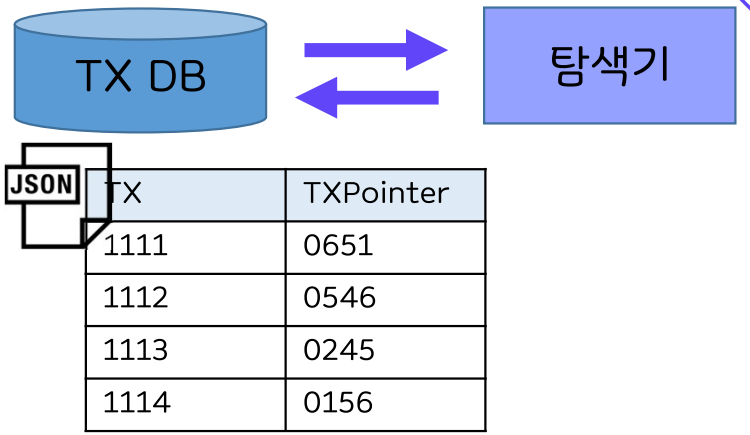
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

탐색기 구현
JSON 파일 싹 불러와서
TXPointer찾은 후, 맞는 TX값 반환

개발 및 구현

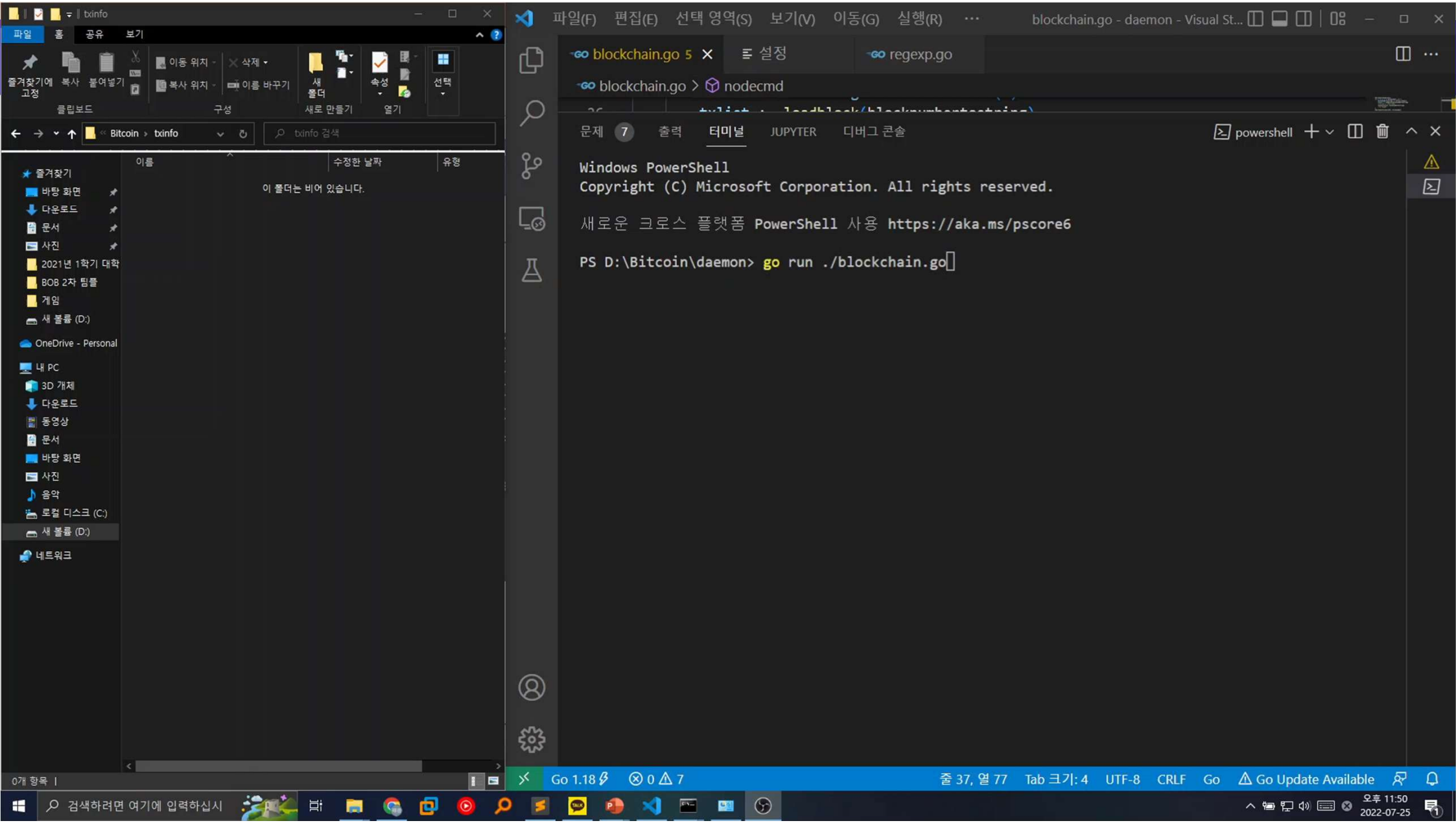
```
fmt.Println("search tx info : " + searchtxinfo)
fmt.Println("next tx info :")
fmt.Println(list)
fmt.Println("")
if list != nil {
    for i := 0; i < len(list); i++ {
        searchnexttx(list[i], blocknumber)
    }

}
```



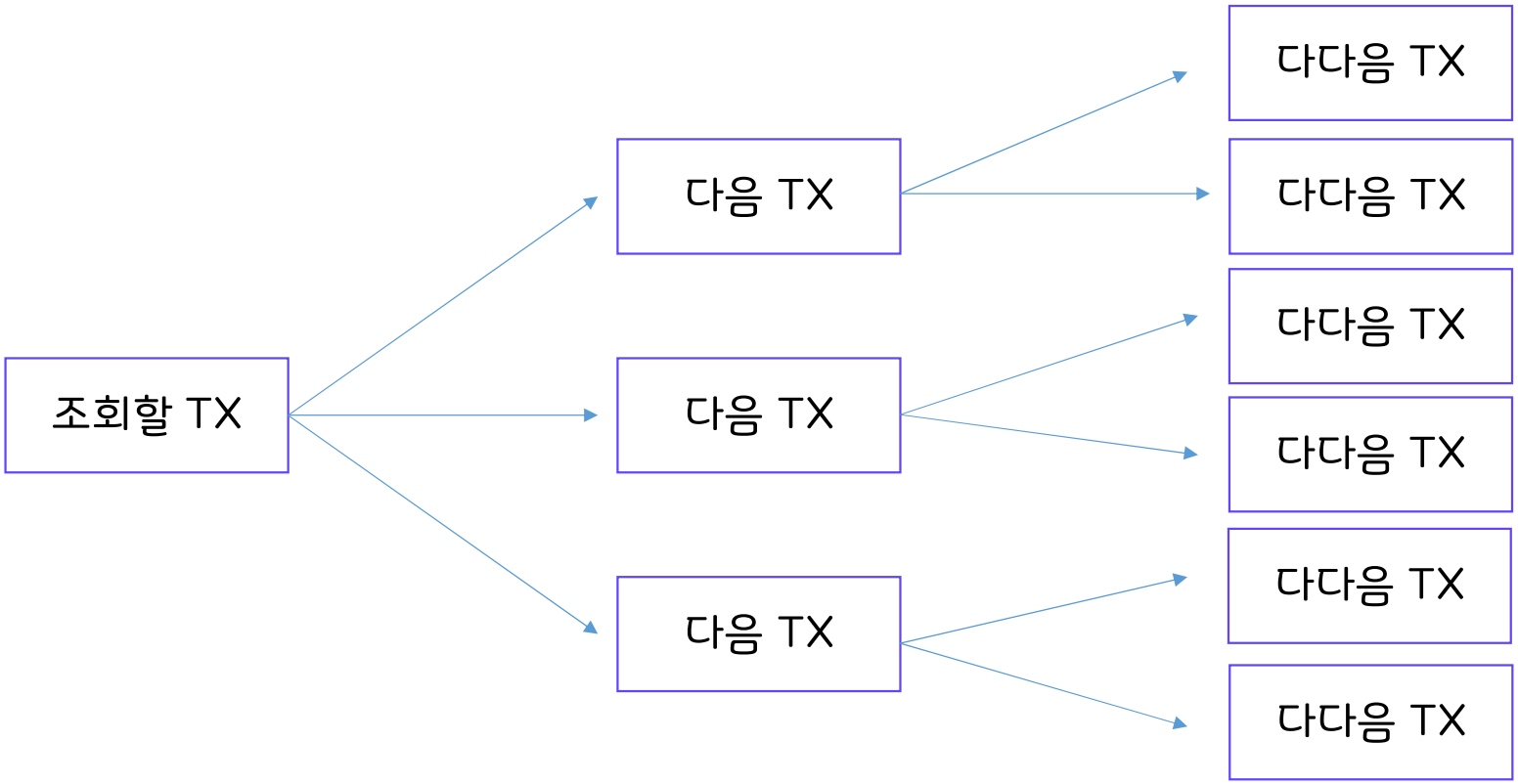
TX	TXPointer
1111	0651
1112	0546
1113	0245
1114	0156

재귀함수 넣어서
전부 찾는걸로 구현

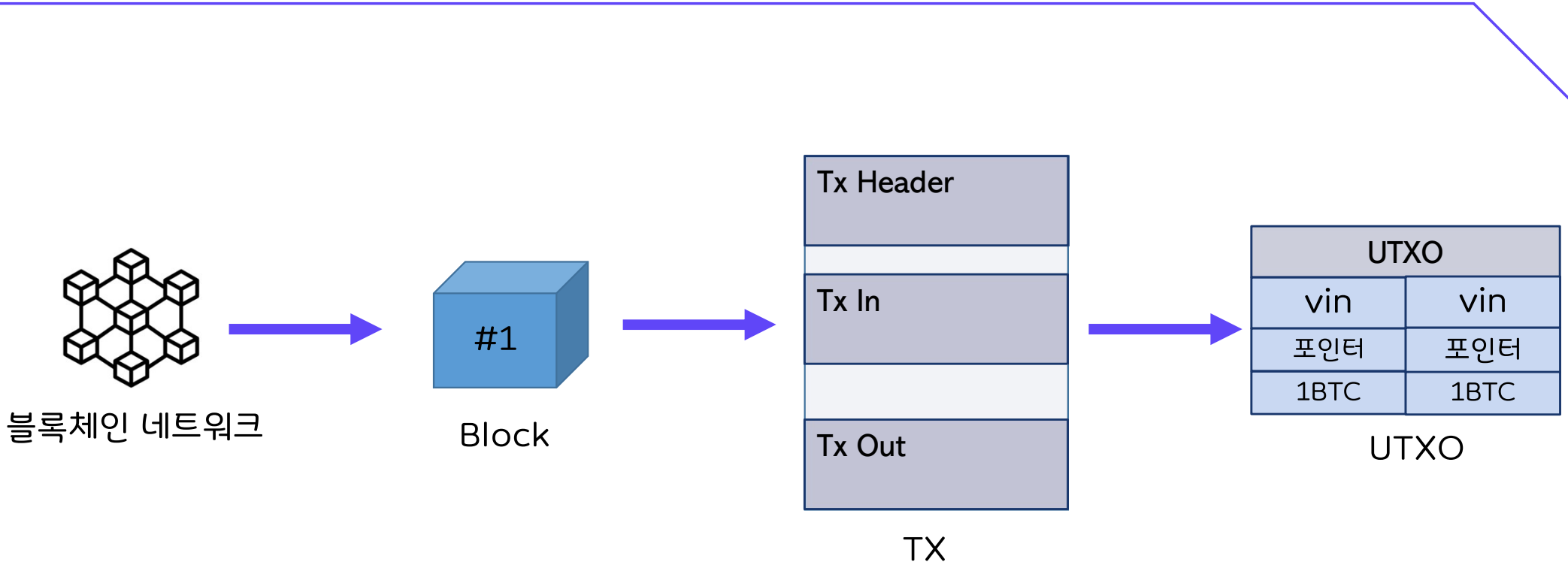

txinfo
파일 홈 공유 보기
즐겨찾기에 고정
복사
붙여넣기
이동 위치
복사 위치
삭제
이름 바꾸기
새 폴더
속성
선택
클립보드
구성
새로 만들기
열기
Bitcoin › txinfo
txinfo 검색
이름
수정한 날짜
유형
이 폴더는 비어 있습니다.
즐겨찾기
바탕 화면
다운로드
문서
사진
2021년 1학기 대학
BOB 2차 팀플
게임
새 볼륨 (D:)
OneDrive - Personal
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
새 볼륨 (D:)
네트워크
0개 항목
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) ···
blockchain.go - daemon - Visual St...
blockchain.go 5
설정
regexp.go
blockchain.go > nodecmd
문제 7
출력
터미널
JUPYTER
디버그 콘솔
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6
PS D:\Bitcoin\daemon> go run ./blockchain.go
Go 1.18
0 7
줄 37, 열 77
Tab 크기: 4
UTF-8
CRLF
Go
Go Update Available
검색하려면 여기에 입력하십시
오후 11:50
2022-07-25

04 후기

후기



후기


블록체인 네트워크
#1
Block
Tx Header
Tx In
Tx Out
TX
UTXO
vin
vin
포인터
포인터
1BTC
1BTC
UTXO

Q & A

끝 감사합니다:)

91714167 유재경