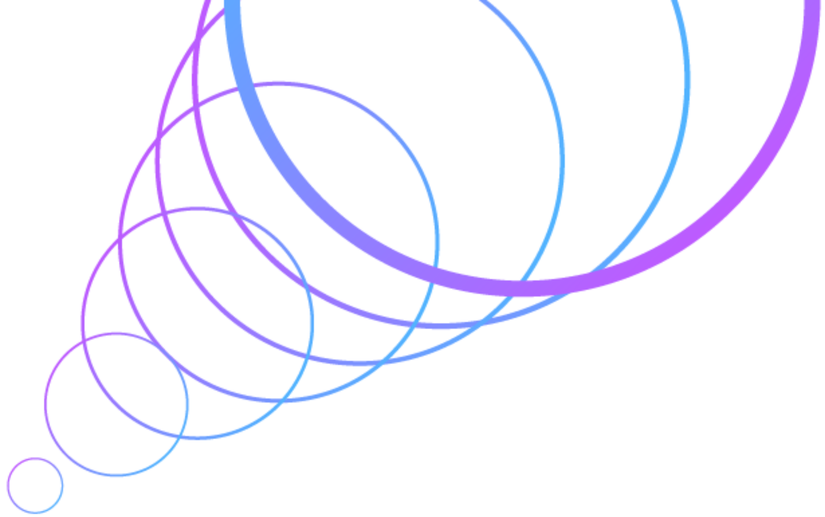
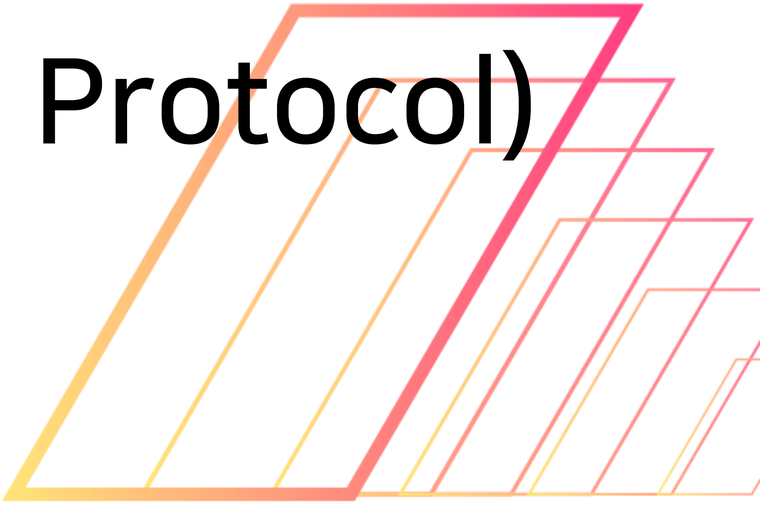




NAVER MongoDB 접근통제 개발기
(feat. Electron & Mongo Wire Protocol)

박경덕 NAVER CLOUD

CONTENTS

세션에서 전달하고자 하는 내용

DEVIEW 2021

다루는 내용

- MongoDB 접근통제 시스템을 구축하면서 마주한 이슈들을 어떻게 풀어갔는지
- 데스크탑 앱으로 Electron을 사용한 이유와 구현간 이슈 해결
- MongoDB Gateway 역할을 어떻게 구현했는지

다루는 않는내용

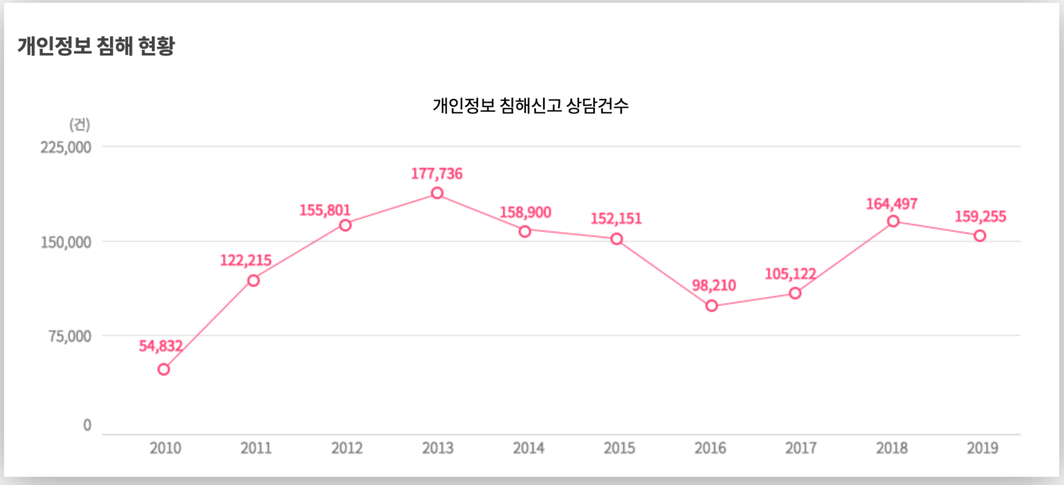
- Electron 구현간 사용한 API 설명
- Wire Protocol을 구현한 상세 소스코드

1.NAVER DB 접근제어 시스템

1.1 DB 접근제어 시스템을 왜 사용하는가

DEVIEW 2021

줄어들지 않는 정보 유출 사건



	2015	2016	2017	2018	2019
합계	152,151	98,210	105,122	164,497	159,255
- 개인정보 무단수집	2,442	2,568	1,876	2,764	3,237
- 개인정보 무단이용제공	3,585	3,141	3,881	6,457	6,055
- 주민번호등 타인정보도용	77,598	48,557	63,189	111,483	134,271
- 회원탈퇴 또는 정정 요구 불응	957	855	862	1,149	1,292
- 법적용 불가 침해사례	60,480	38,239	30,972	37,156	8,745
- 기타	7,089	4,850	4,342	5,488	5,655

https://www.privacy.go.kr/nns/ntc/pex/personalExam.do

1.2 DBIP

서비스 중인 DBMS의 형상과 권한을 관리하는 시스템

- DB, Table, Column 및 개인정보 등 형상관리
- 사용자 DB접근 권한 및 상세 권한(DML, DDL) 관리

DB 형상

Authorization

Authentication

접속이력 / Query Log

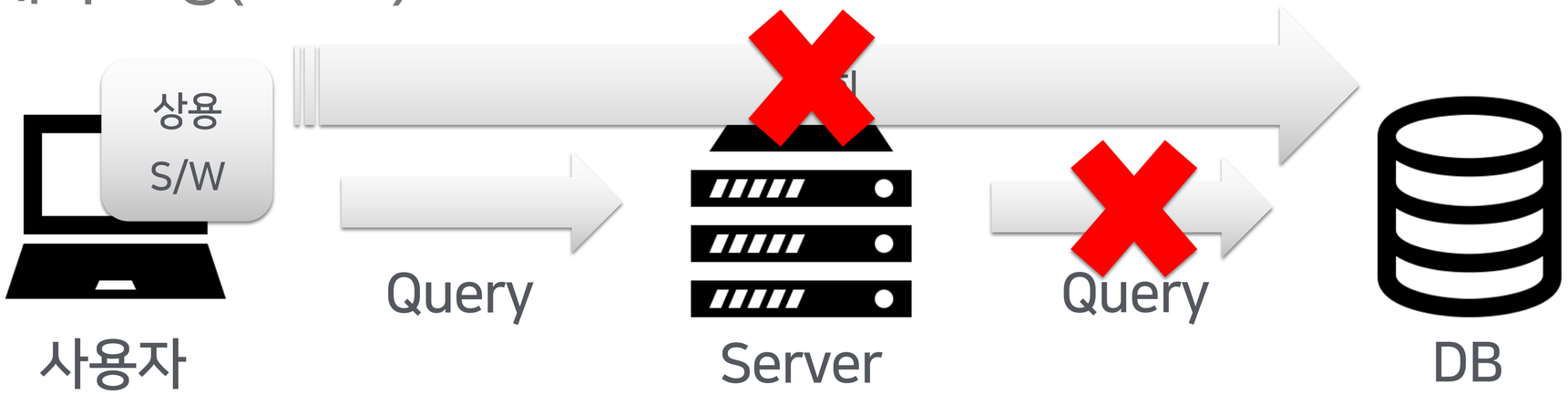
DB ACL

표준단어/도메인

1.2 SQL Gateway

DB 접근 통제 시스템

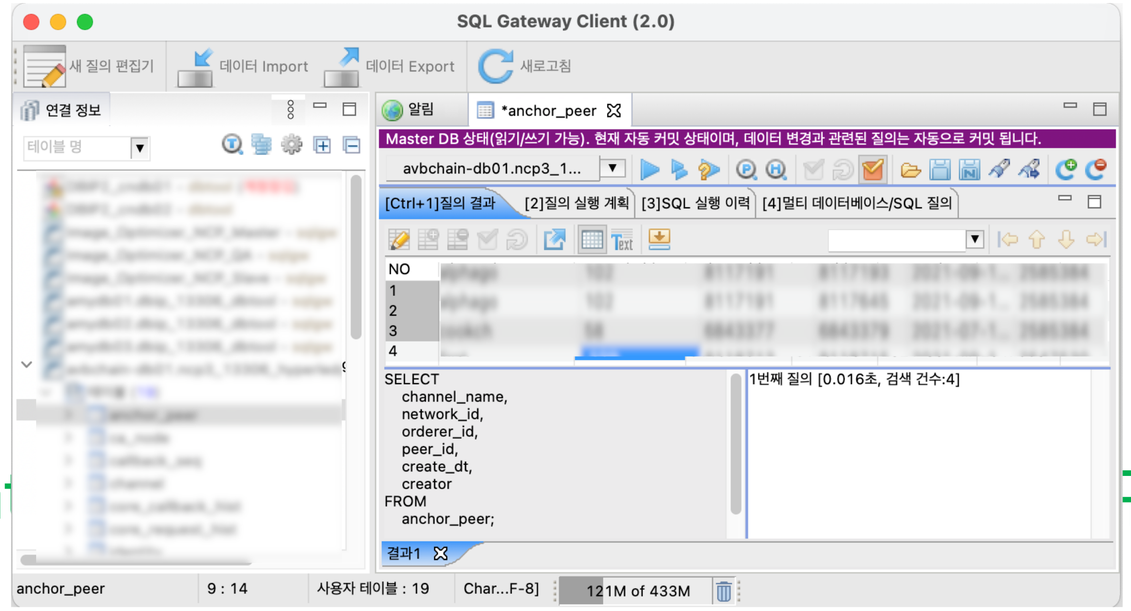
- 개발자가 서비스 중인 DB에 접근하여 Query 하는 것을 권한에 따라 통제
- 상세권한종류 : SELECT, INSERT, UPDATE, DELETE, DDL, EXCUTE, timeout, 조회건수 등
- 실행제한 : 개인/중요정보 열람/반출 제한, 접속환경에 따른 제약 등
- Query 내역 로깅(audit)



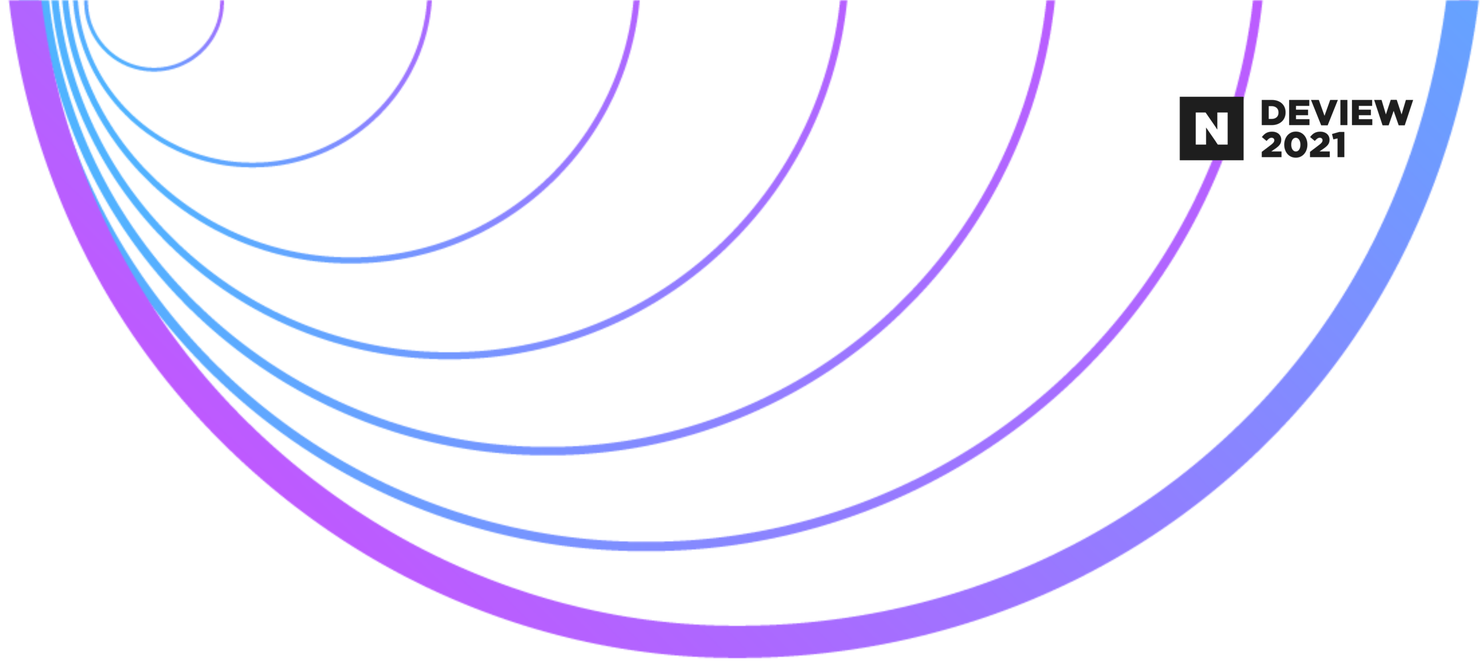
1.3 MongoDB SQL Gateway

새로운 SQL Gateway 개발필요성 대두

- MongoDB도 접근통제 적용이 필요함
- SQL Gateway는 RDB 구조에 정형화되어 NoSQL DB는 사용이 어려움
- MongoDB는 JDBC spec을 구현하지 않고 자체 프로토콜을 사용함



SQL Ga 구 개발

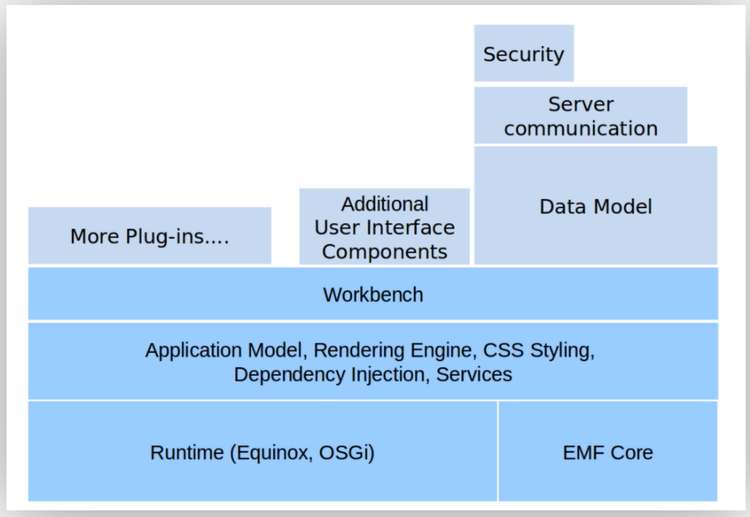


2. Client (Electron)

2.1 기존 SQL Gateway Client 의 한계

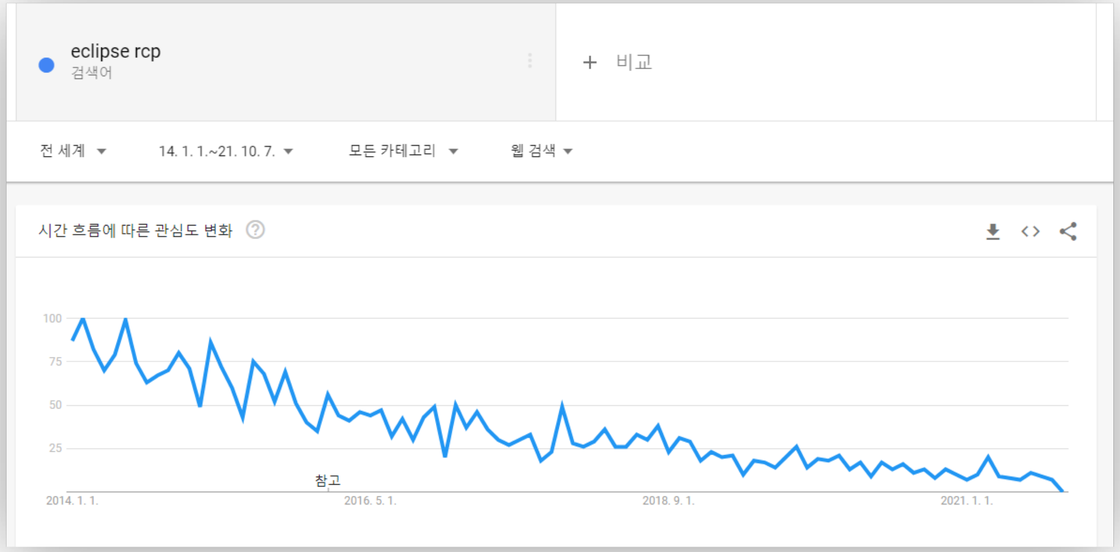
기존 Client의 노후화

- Eclipse RCP(Rich Client Platform) 기반
- 러닝커브가 높아 유지보수가 어려움
- 활용되는 곳이 많지 않아 개발 관련 참고 자료가 부족



RCP architecture

https://www.vogella.com/tutorials/EclipseRCP/article.html



구글 트렌드 변화량

2.2 새로운 플랫폼 선정

Desktop Application vs Web

비교항목	Desktop Application	WEB
Connection 유지	TCP heartbeat	Web socket
저장공간	메모리, 파일 가능	로컬스토리지(최대10M), 세션 스토리지(5M)
대용량 데이터 처리	TCP Socket으로 대용량 문제 없음	HTTP/Web socket 용량 한계
복사방지	가능	가능하지만 해제가능
배포방식	사용자가 Client App Download	Server에만 배포

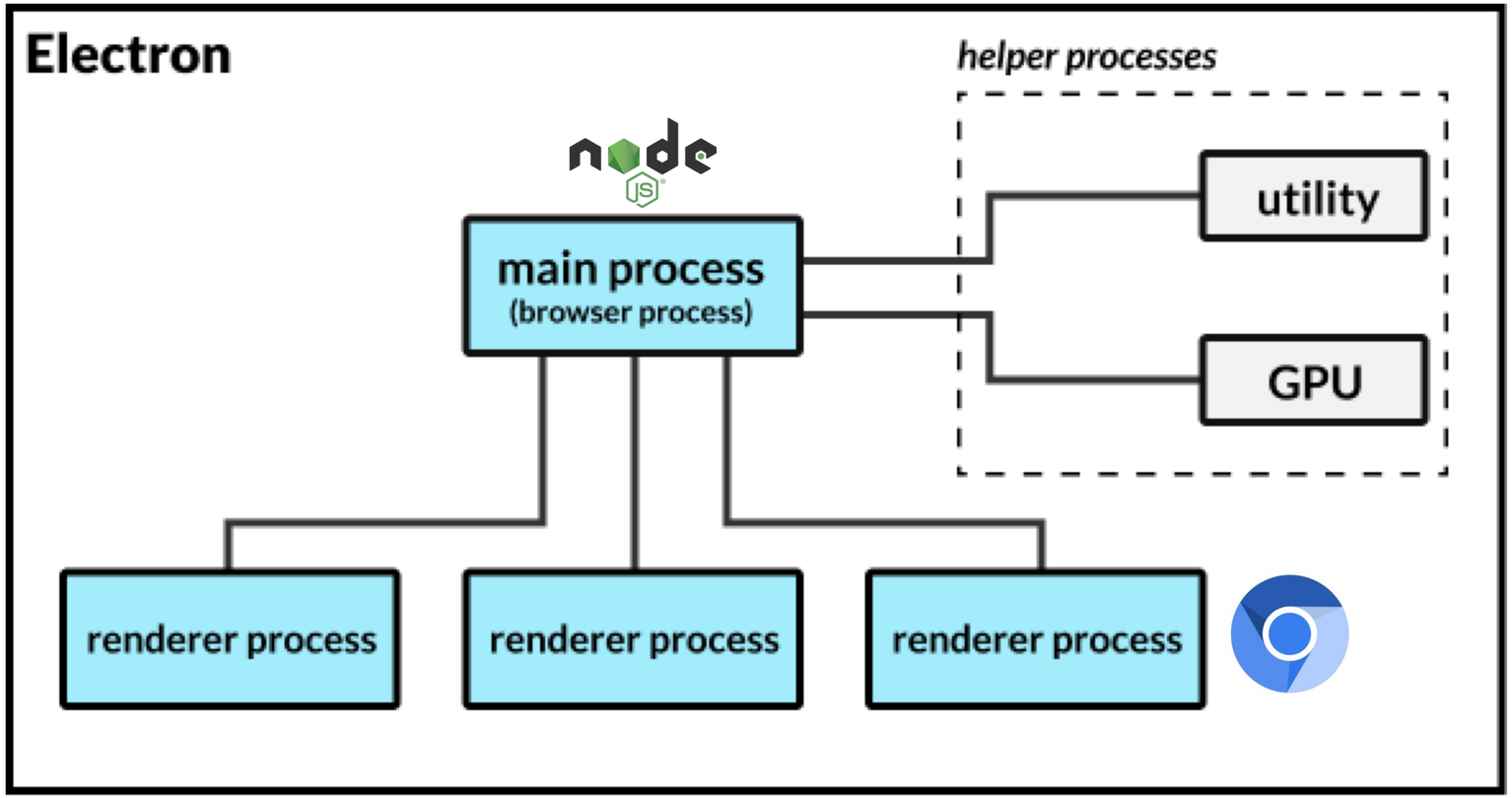
2.3.2 Electron 선택

많은 장점을 가진 Electron을 선택

- Web기반 기술을 이용한 접근성(JavaScript, CSS, HTML)
- 오픈 소스로 운영되어 라이브러리가 많고 커뮤니티가 활성화 되었음
- Mac, Windows, Linux 크로스 플랫폼 지원

2.3.3 Electron 구조

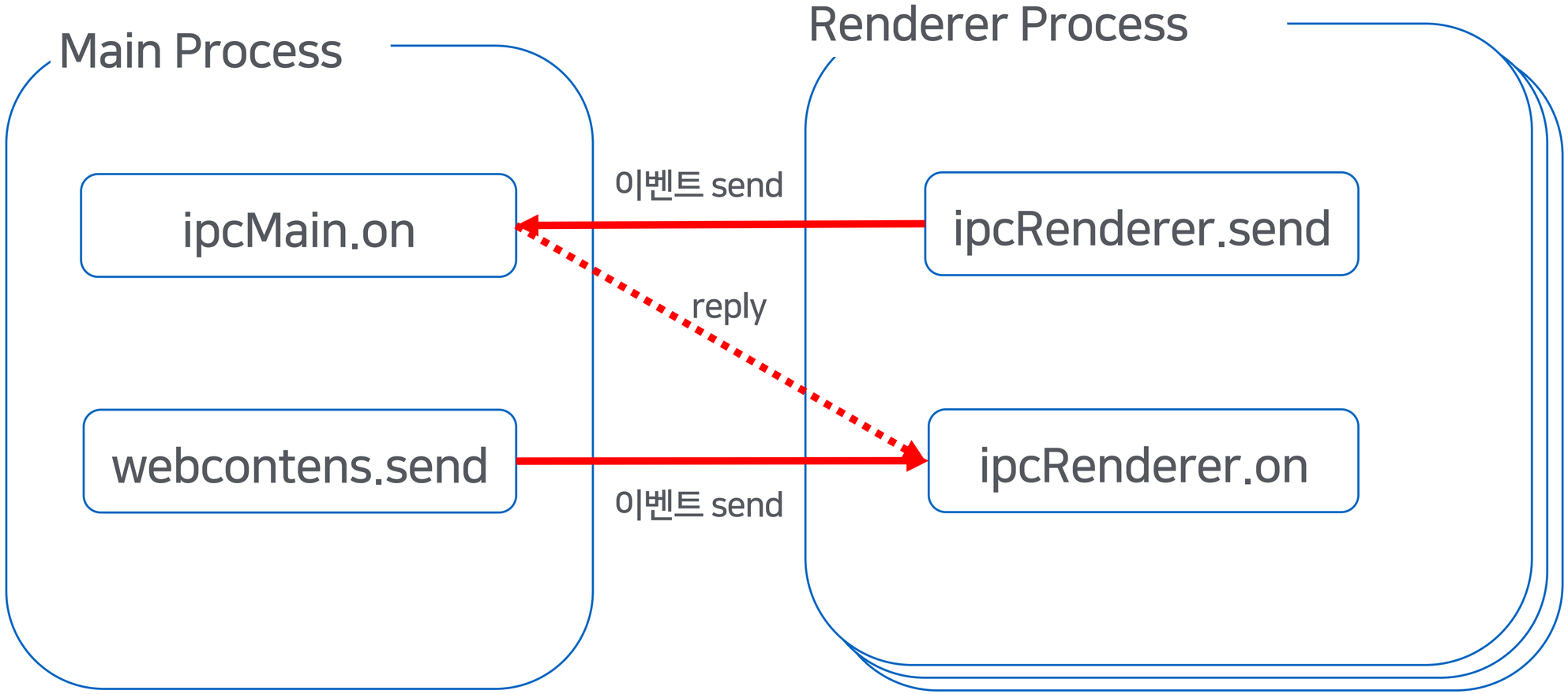
Main Process와 Renderer Process 가 분리



2.3.4 Electron Process간 통신

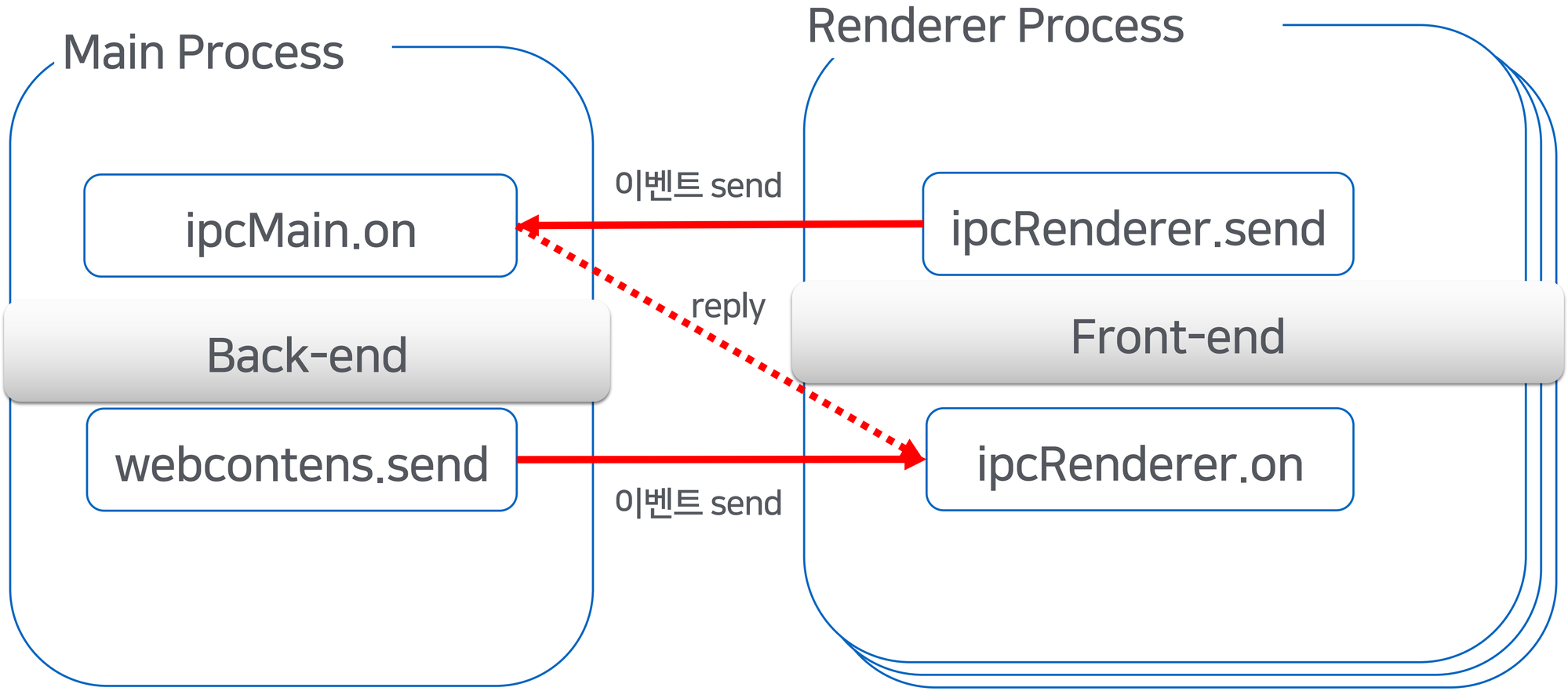
DEVIEW 2021

IPC(Inter Process Communication) 모듈로 통신



2.3.4 Electron Process간 통신

WEB 과 비교해본다면

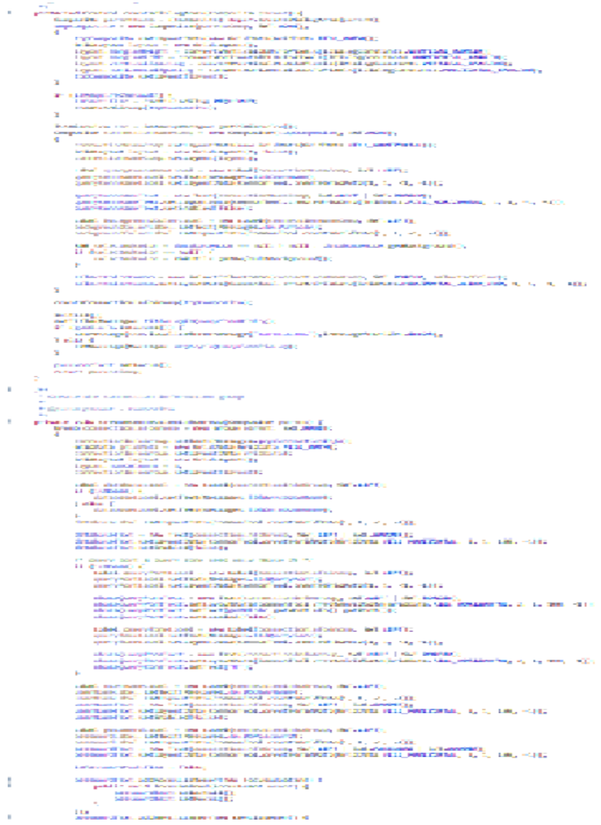


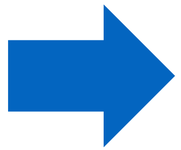
2.4.1 Electron 을 사용하며 얻은 이점

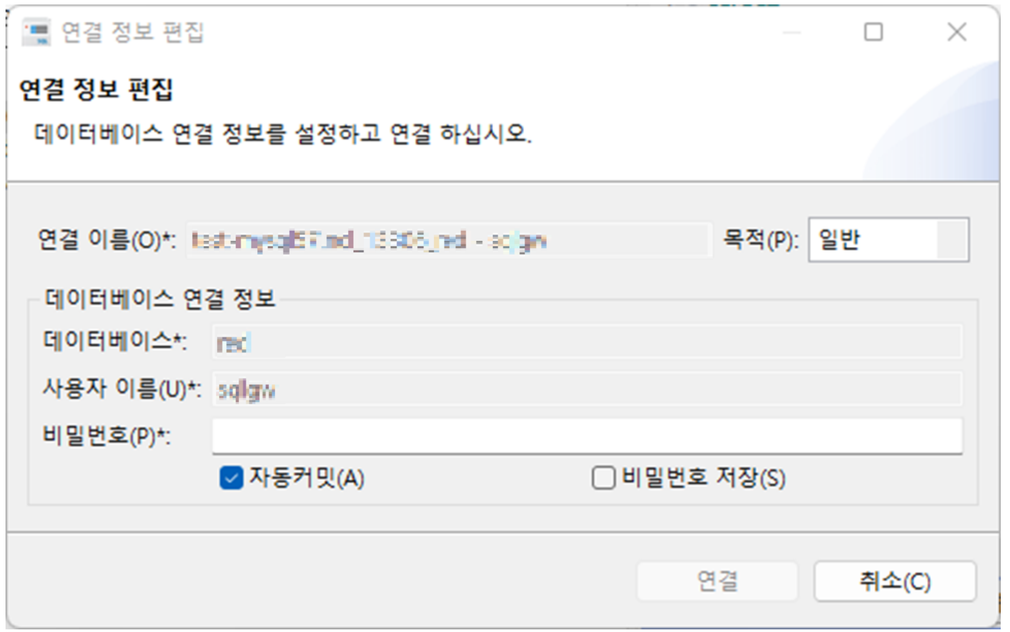
DEVIEW 2021

WEB 기반 기술로 개발하여 소스 코드 간결화 및 유지보수 용이

- 기존 RCP의 장황하게 긴 UI 코드







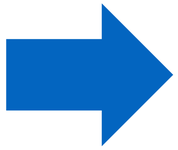
2.4.1 Electron 을 사용하며 얻은 이점

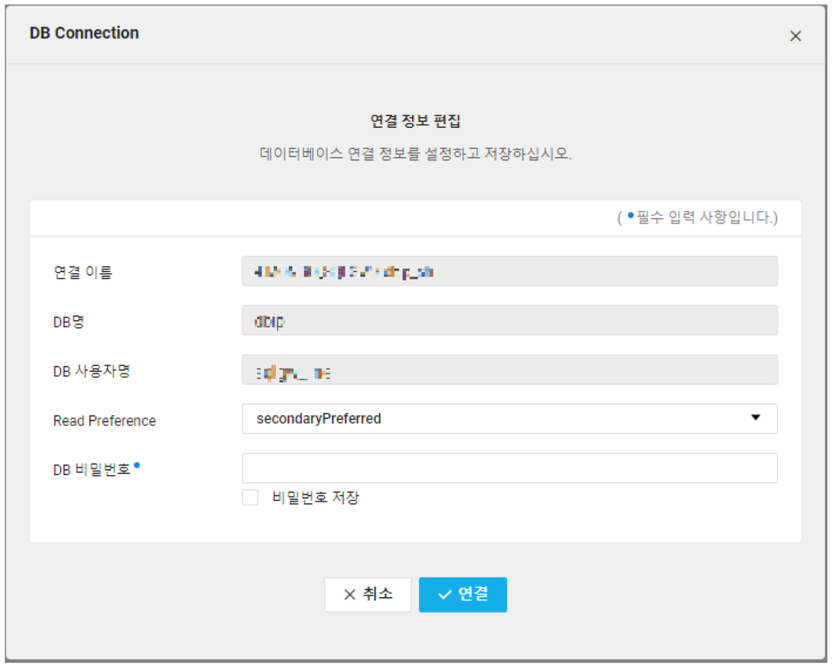
DEVIEW 2021

Web 기반 기술로개발하여 소스 코드 간결화 및 유지보수 용이

- Web 기반의 간결한 코드

```
<div class="box mt-30">
  <div class "box-header text-right text-888 pv-6">
    ( <span class="ess ex"></span>{{ $t("validation.required') }})
  </div>
  <div class="box-body">
    <div class="form-group row">
      <label class "form-control-label col-3">
        <span>{{ $t('conn.connName') }}</span>
      </lubul>
      <div class="col-9">
        <input v-model="dbConnForm.connName" type="text" class="form-control" disabled />
      </div>
    </div>
    <div class="form group row">
      <label class "form-control-label col-3">
        <span>{{ $t('db.dbName') }}</span>
      </lubul>
      <div class "col-9">
        <input v model="dbConnForm.dbName" class="form control" disabled />
      </div>
    </div>

    <div class "form-group row">
      <label class="form-control-label col-4">
        <span>{{ $t('db.dbUser') }}</span>
      </label>
      <div class="col 9">
        <input v-model "dbConnForm.dbUser" type "text" class "form-control" disabled />
      </div>
    </div>
```

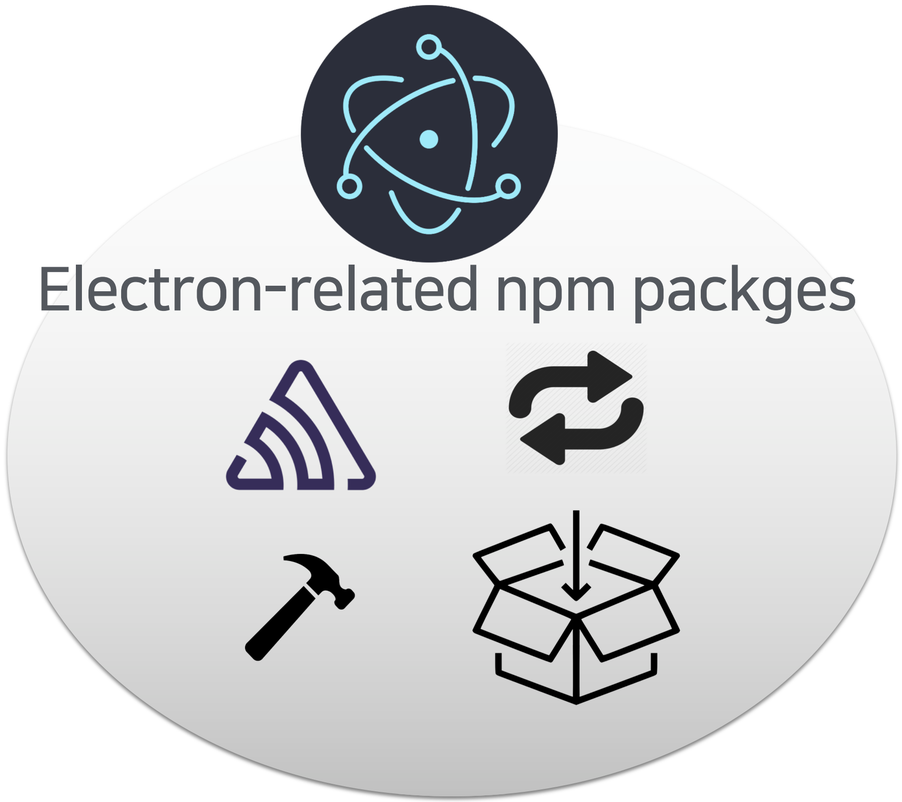




2.4.2 Electron 을 사용하며 얻은 이점

Desktop Application 관련 다양한 Library 사용 가능

- 자동 업데이트 : Electron-updater
- 플랫폼별 설치 관리 : Electron-builder
- 에러 관리 : Electron-Sentry SDK
- 패키징 : Electron-forge



https://www.electronjs.org/userland/most_downloaded_packages

2.5.1 Electron을 사용시 유의할 점

DEVIEW 2021

소스코드 보안 취약

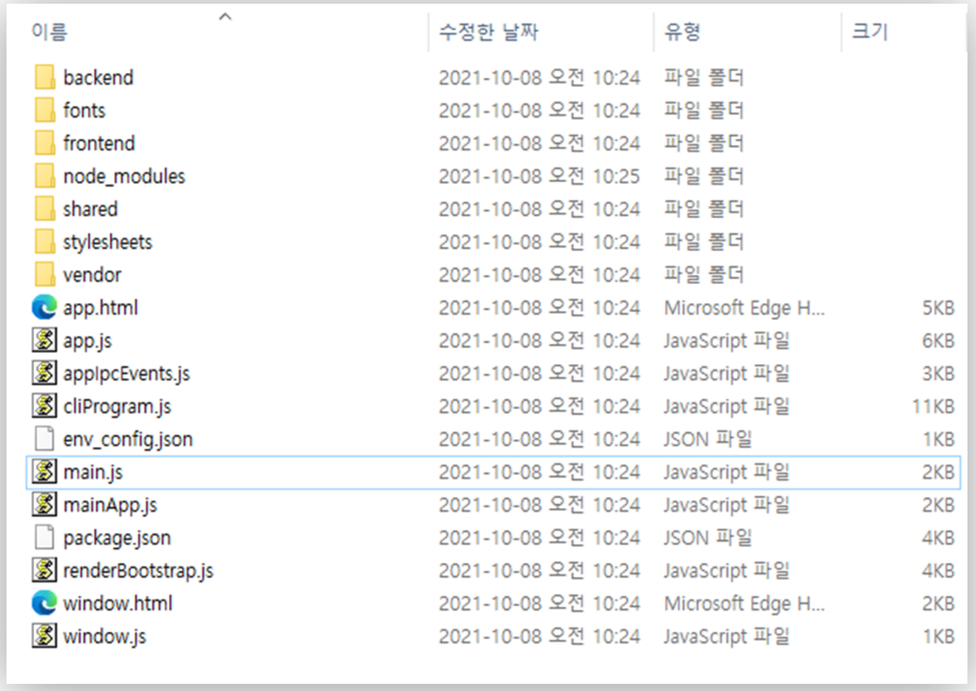
- 소스코드를 asar형식으로 패키징 하기 때문에 unpack 하여 소스코드를 확인할 수 있음
- 때문에 소스를 수정하여 악의적 공격 가능

Extract the whole archive:

```
npx asar extract app.asar destfolder
```

unpackage

unpack



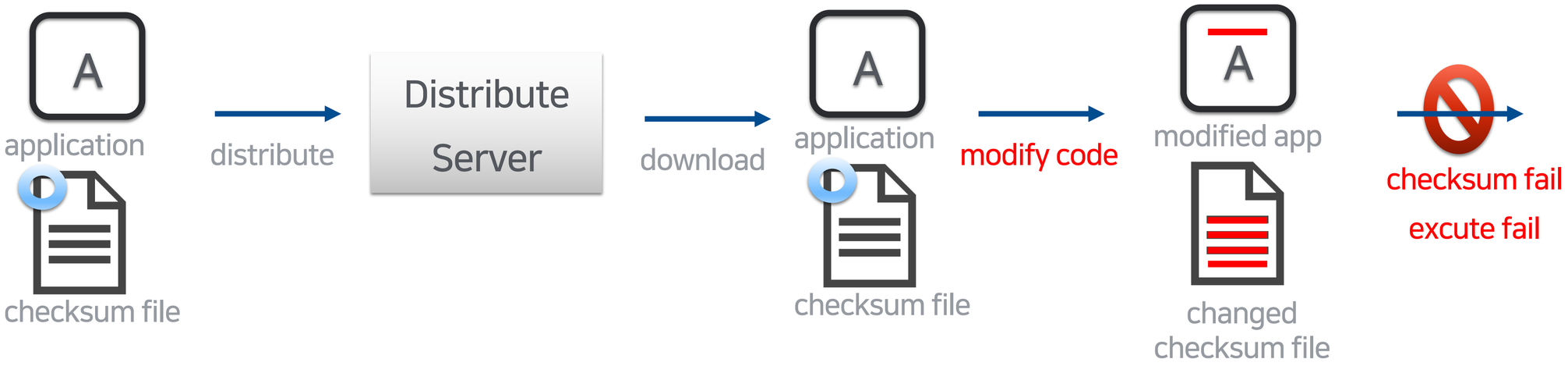
js등 소스코드 공개됨

2.5.2 Electron을 사용시 유의할 점

DEVIEW 2021

소스코드 보안 취약 방지 방안

- Checksum 확인 기능을 추가하여 소스코드 위변조 방지
- 소스코드 난독화 기능 추가

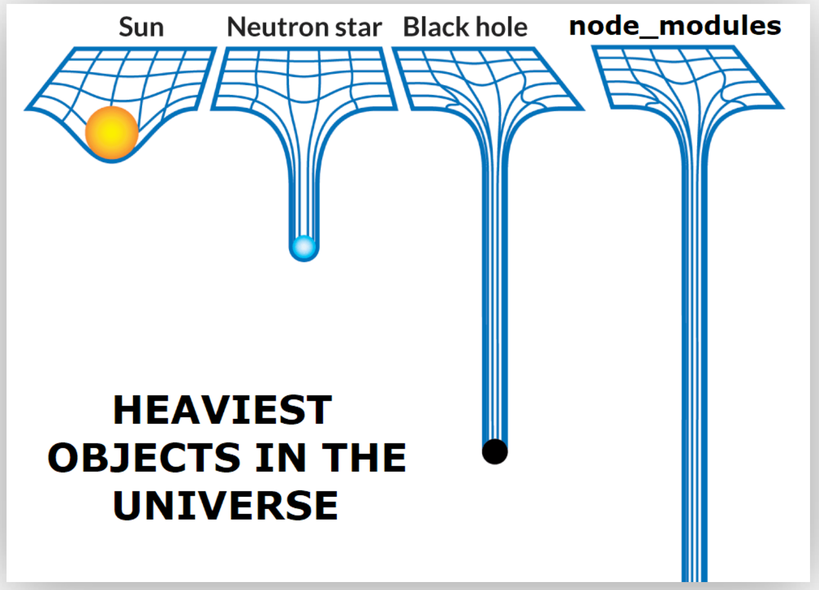


2.5.3 Electron을 사용시 유의할 점

DEVIEW 2021

상대적으로 무거움

- Chromium 기반으로 동작하는 Electron
- 간단한 프로그램을 만들어도 Native Code들 보다 무겁고 메모리 사용이 많음



node module의 무거움…

https://steemkr.com/nodejs/@shivayl/the-heaviest-objects-in-the-universe-20171010t203724433z

2.6 Client 구현 Point

1. Client 구조

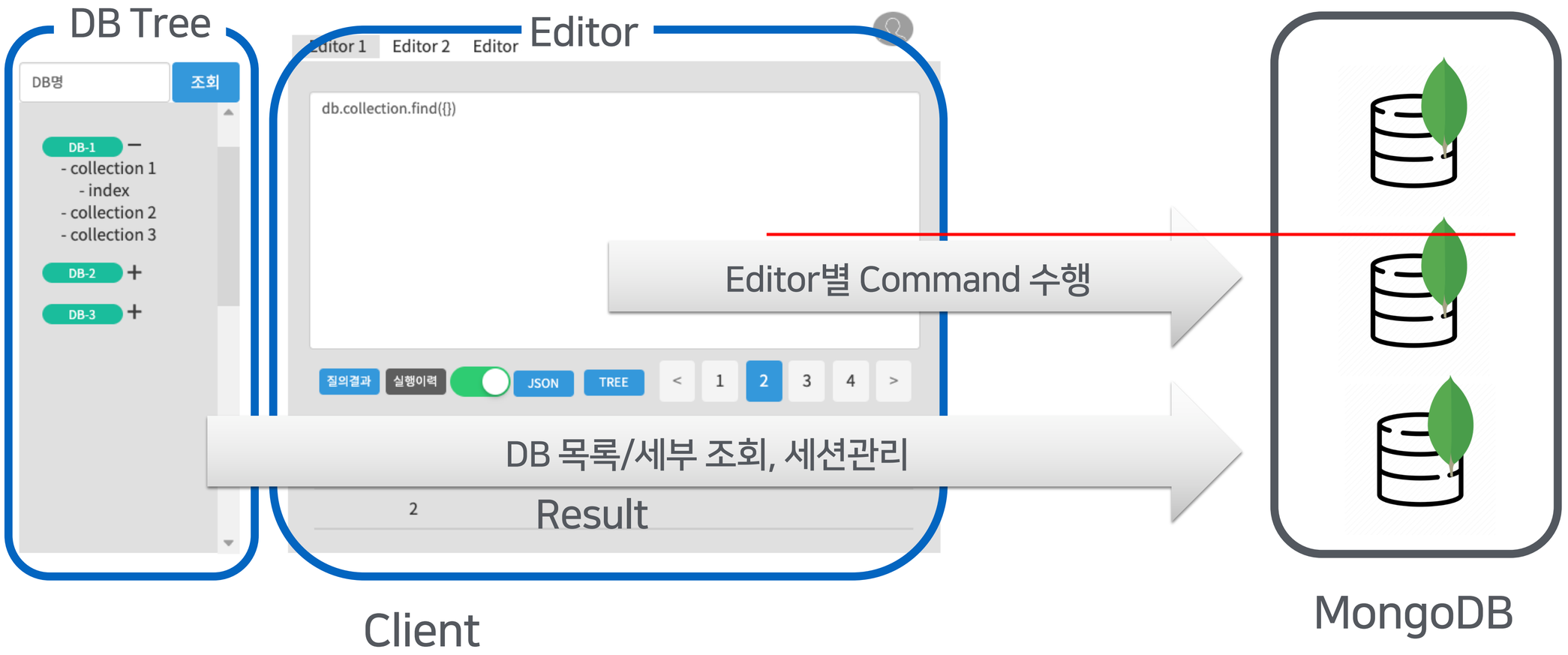
2. MongoDB Command 구현

3. Import / Export Stream 으로 구현

4. 다중 Command 이슈

2.6.1 Client 구조

DB tree, Editor



2.6.2 MongoDB Command 구현

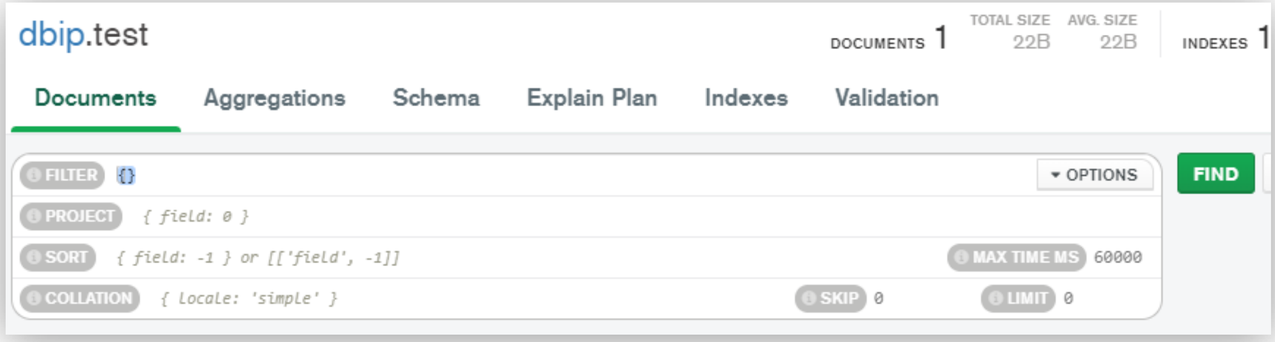
DEVIEW 2021

Command 구현 방식 선택

- 직접 Command를 Editor에 입력하는 방식 (CLI)

선택

- Command 타입/조건/정렬 등을 입력을 Assist하는 방식 (UI)



2.6.2 MongoDB Command 구현

DEVIEW 2021

Command 변환

- Command로 node.js 코드로 변경 해야함

DB Name : nflix

collection name : dramas

title	company
Squid Game	A
Kindom	B

```
use nflix
db.dramas.findOne({title : "Squid Game"})
```

사용자가 command 입력

```
await client.connect();
const database = client.db("nflix");
const dramas = database.collection("dramas");
const query = {title : "Squid Game"};
const drama = await dramas.findOne(query);
```

javascript 변환

Command 별로 구현을 한다고 하더라도
유지보수는…?

2.6.2 MongoDB Command 구현

DEVIEW 2021

Proxy모듈을 통한 해결 방안

- Proxy를 통한 코드 간결화
- 공식 Driver 활용 가능

1. 사용자 Command 입력

```
db.dramas.findOne({title : "Squid Game"})
```

2. DB 연결

```
await client.connect();
const database = client.db("nflix");
```

2.6.2 MongoDB Command 구현

Proxy모듈을 통한 해결 방안

3. Proxy 모듈에서 collection 객체를 리턴

```
// WrappedMongoDb.ts
export default function(mongoDb:Db){
      let classHandler = {
        get: function(target:Db, prop:string) {
            const ret = Reflect.get(target, prop);
            // database 객체에있던 함수 호출 할수 있도록 함
            if (prop in target) {
                return ret;
            } else {
            // database에 있던 함수가 아닐경우 collection명을 호출하도록함.
                return target.collection(prop);
            }
        }
    }
    return new Proxy(mongoDb, classHandler);
}
```

4. eval 로 명령어 수행

```
this.resultset = eval(query);
```

참조 : https://github.com/yonn-kim/mongo-js-driver-wrapper

2.6.2 MongoDB Command 구현

Proxy모듈 대신 Mongo shell 사용

- Mongo DB 커맨드를 수행하는 공식 지원 CLI 툴 (https://docs.mongodb.com/mongodb-shell)
- 커스텀 개발이 가능한 오픈소스 (https://github.com/mongodb-js/mongosh)

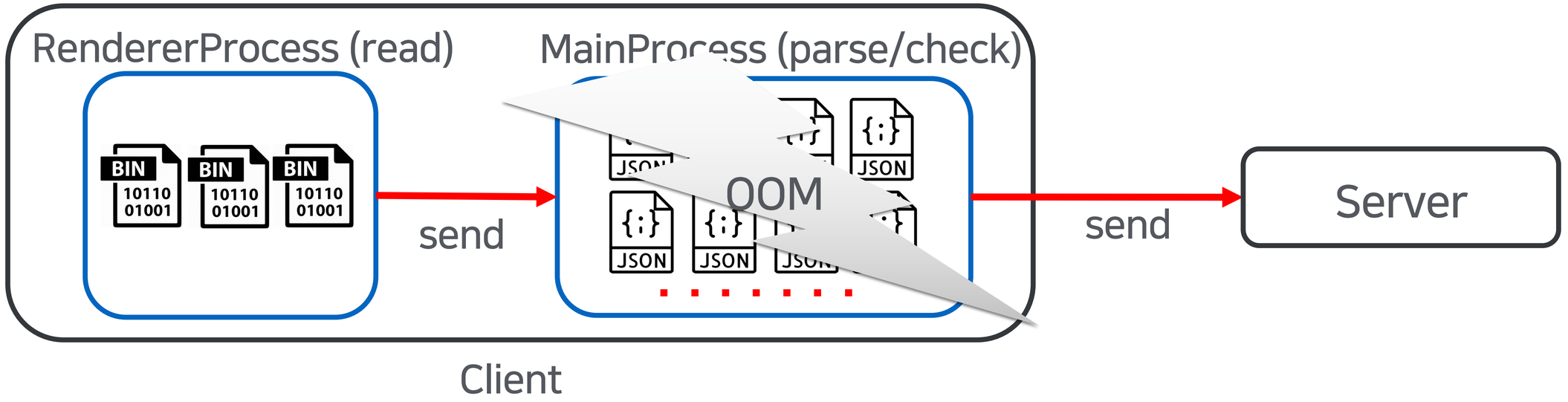
```
mongodb@ubuntu:~/mongodb-linux-x86_64-2.6.0$ bin/mongo
MongoDB shell version: 2.6.0
connecting to: test
> db.adminCommand( { getLog: "global" } )
{
  "totalLinesWritten" : 34,
  "log" : [
    "2014-05-08T01:36:03.034-0400 [initandlisten] MongoDB ... ",
    "2014-05-08T01:36:03.036-0400 [initandlisten] db version v2.6.0",
    "2014-05-08T01:36:03.038-0400 [initandlisten] git version: ... ",
    ...
  ],
  "ok" : 1
}
>
```

Mongo Shell CLI 화면

2.6.3 Data Import / Export 구현

대용량 데이터 처리시 이슈

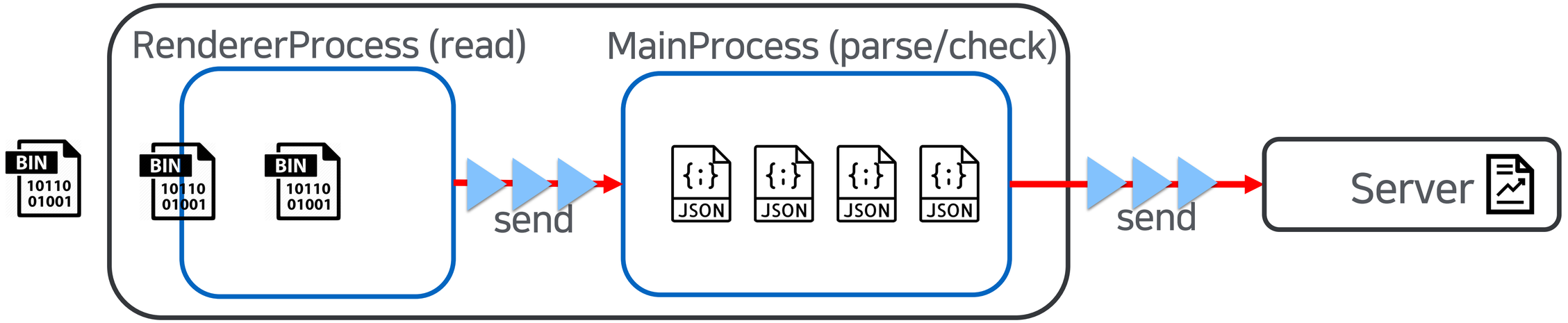
- Renderer에서는 데이터를 MainProcess으로 보내서 처리 해야함
- 그러나 MainProcess에서는 Multi Thread 를 권장하지 않음
- 대용량 데이터를 한번에 처리하려면 OOM 발생 -> Chunk 단위로 처리 필요



2.6.3 Data Import / Export 구현

Stream 처리

- Stream모듈을 구현하여 병목구간 해결
- Chunk 단위로 처리하여 OOM 방지
- 비동기로 데이터를 I/O 처리



2.6.3 Data Import Stream 구현 예제

DEVIEW 2021

Import Stream 처리

```
const jsonToMqlStream = pipeline(
      fs.createReadStream(filePath, { encoding: 'utf8' }),
      JSONStream.parse('*'),
      new ImportJsonQuery(req).on('data', (res: ExecuteQueryInfo) => {
        result.push(res)
      }),
      err => {
        if (err) {
          // JSON parsing error
          errorMsg = 'Json Parsing error \n' + err.message
        }
      },
    )
```

Read file stream

```
const importStream = pipeline(
      new ExecQueryReadStream(infoList),
      new QueryExecutor(req, this.queryRunningPubsub),
      new ImportReport(editorId).on('data', (res: ImportResult) => {
        result.push(res)
      }),
      err => {
        if (err) {
          errorMsg = 'Import mql error \n' + err.message
          Sentry.captureException(err)
        }
      },
    )
```

Import stream

2.6.4 Multi Command 이슈

DEVIEW 2021

Multi Command를 수행할 때 실패 케이스

- 사용자 권한이 DELETE : O, INSERT : X 경우 아래 Command를 모두 실행

```
db.collection1.delete()                                    //권한 O

db.collection1.insert({"colname" : "sql gateway"})   //권한 X
```

- 함께 수행한 Command들은 에러 발생시 모두 수행이 안되어야 하는데 DELETE 만 수행될 수 있다

Command 처리 간 실수 발생 가능

2.6.4 Multi Command 이슈

코딩이 가능한 커맨드

- 아래 코드 처럼 다양하게 Command가 가능해서 모든 케이스를 잡기 어려움

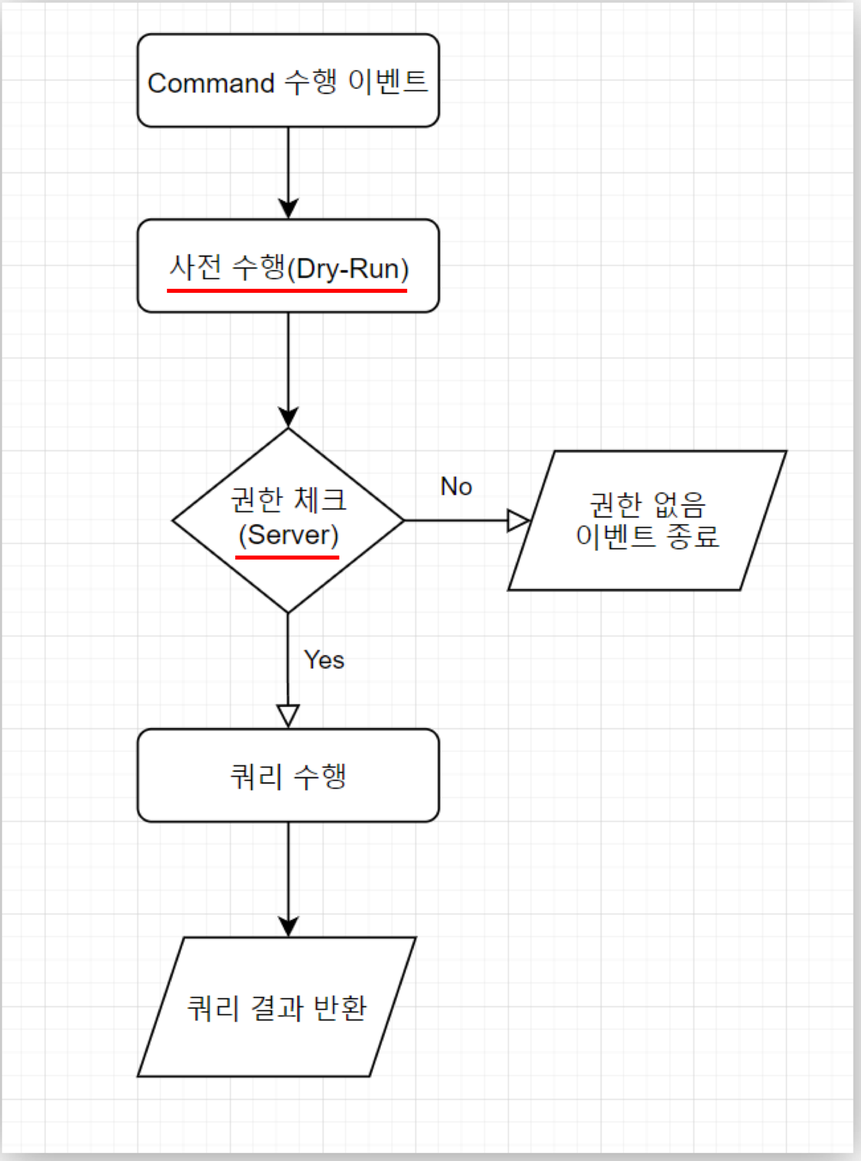
```
function dmlCommand(command, data) {
  db.collection1[command](data)
}

dmlCommand('delete', {title: 'Old'})
dmlCommand('in' + 'sert', {title: 'New'})
```

2.6.4 Multi Command 이슈

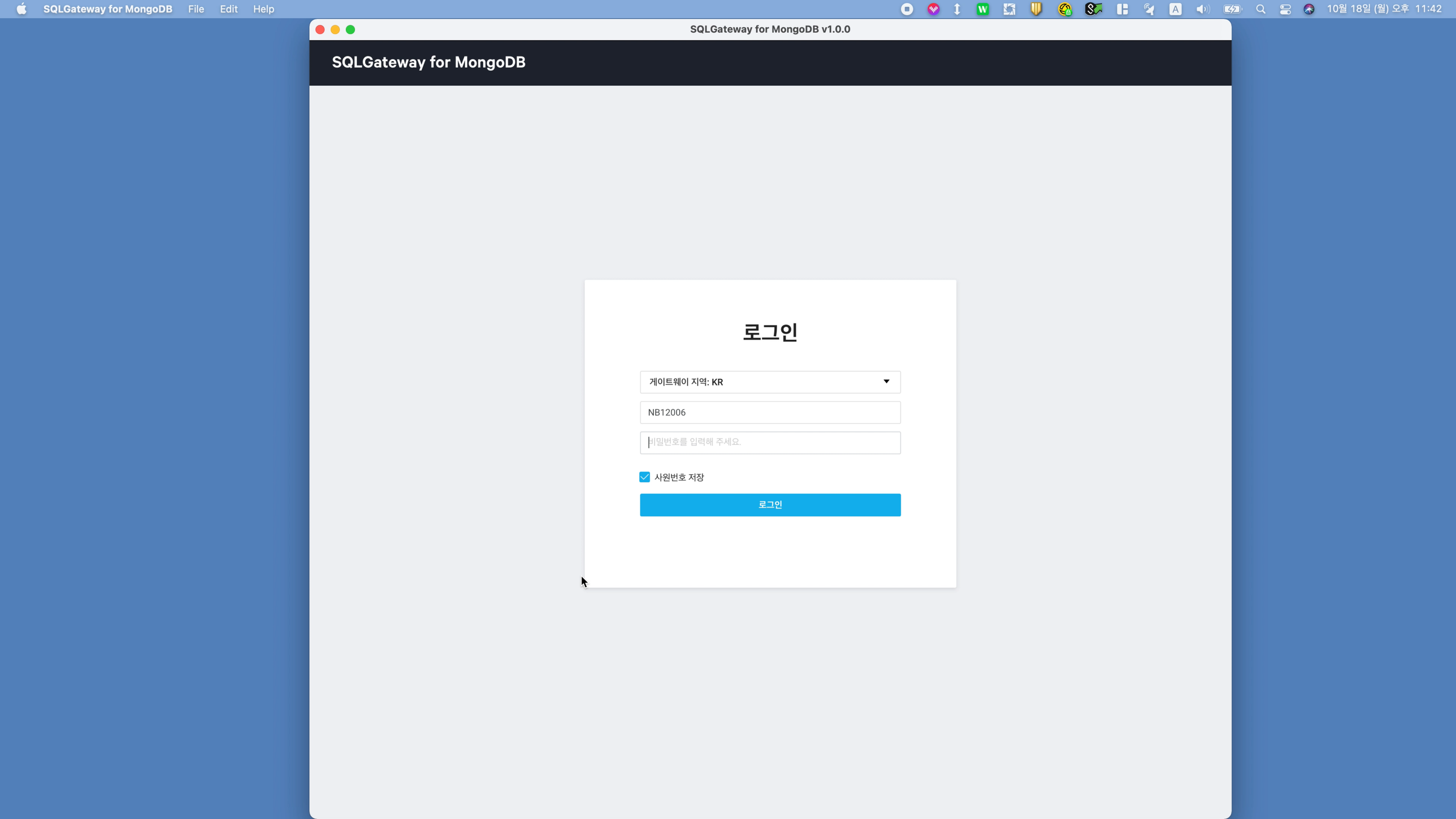
Server에 Command 사전 수행 (dry run)

- Server에서는 Parsing된 Command를 알 수있기 때문에 Command를 체크 할 수 있음

- Server에서 실제 Command를 수행하지 않는 사전 수행으로 Command별 권한 확인 가능



Client 시연


SQLGateway for MongoDB File Edit Help
10월 18일 (월) 오후 11:42
SQLGateway for MongoDB v1.0.0
SQLGateway for MongoDB
로그인
게이트웨이 지역: KR
NB12006
비밀번호를 입력해 주세요.
사원번호 저장
로그인

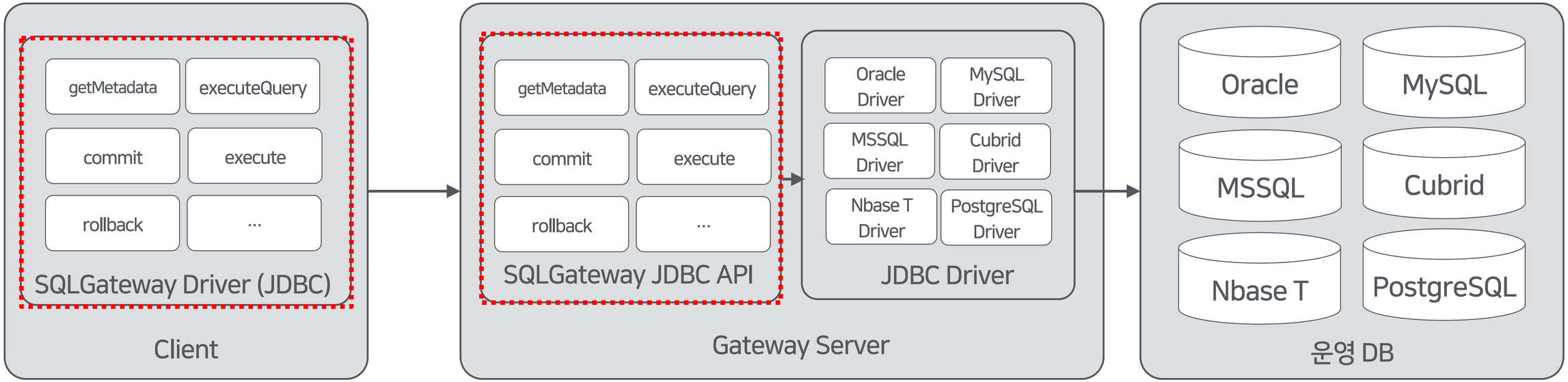
3. Server

(Mongo Wire Protocol & Command)

3.1 기존 SQL Gateway Server

JDBC(Java Database Connectivity) Spec 을 구현한 Gateway Server

- 다양한 DBMS 를 지원하기 위해 JDBC Driver를 사용함
- JDBC spec 을 구현하면 DB 접속툴의 모든 기능 구현 가능



3.2 MongoDB 도 JDBC로?

DEVIEW 2021

기존 SQLGateway로 MongoDB 도 지원할 수 있을까?

- MongoDB 는 JDBC 스펙을 지원 안함
- MongoDB 는 자체 정의 Protocol 사용
- SQLGateway 와 전혀 다른 스펙으로 지원 불가

MongoDB 전용 Gateway Server 구현 필요

3.3 MongoDB 전용 Gateway Server

구현 목표

- 기존 SQLGateway 수준으로 모든 구문 파싱, 접근제어, 로깅 기능을 구현
- MongoDB 기능 변경에 유연한 대응
  - Command 추가, 삭제 또는 옵션 변경
  - MongoDB Protocol 변경

3.3 MongoDB 전용 Gateway Server

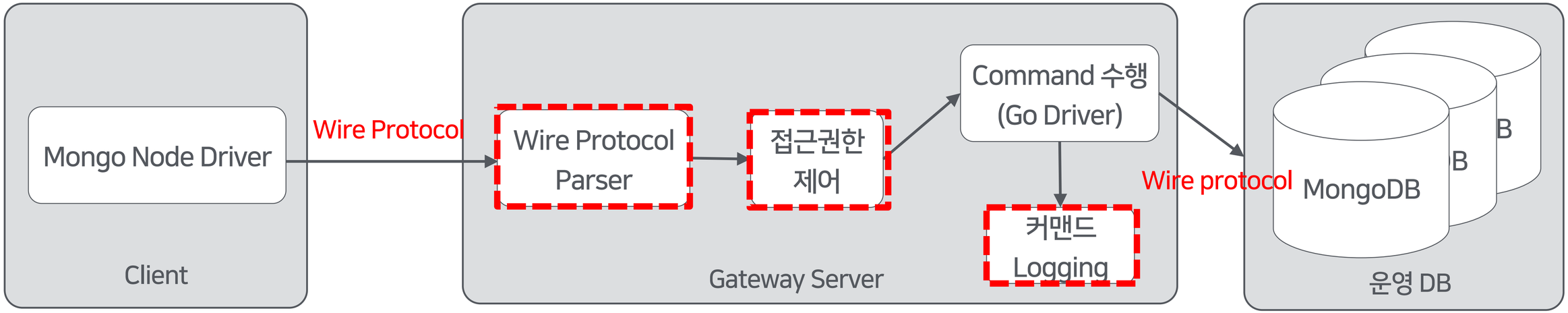
DEVIEW 2021

구현 방법

- Client
  - MongoDB 스펙 변경을 대응을 위해 공식 Driver 그대로 사용
- Server
  - MongoDB Protocol 구현
  - 고성능 네트워크 서버를 쉽게 구현하기 위해 GO 언어로 구현

3.4 MongoDB Gateway 핵심 기능

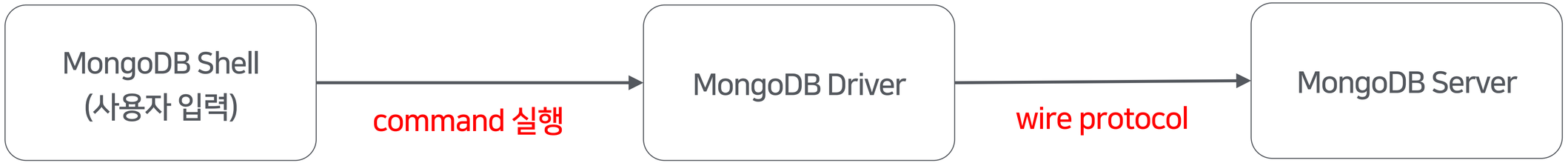
1. MongoDB Wire Protocol Parsing
2. 접근 권한 제어
3. 커맨드 Logging



3.4.1 MongoDB Wire Protocol Parsing

DEVIEW 2021

MongoDB Shell 을 통한 Command 실행 과정



use testdb

db.test.find({})

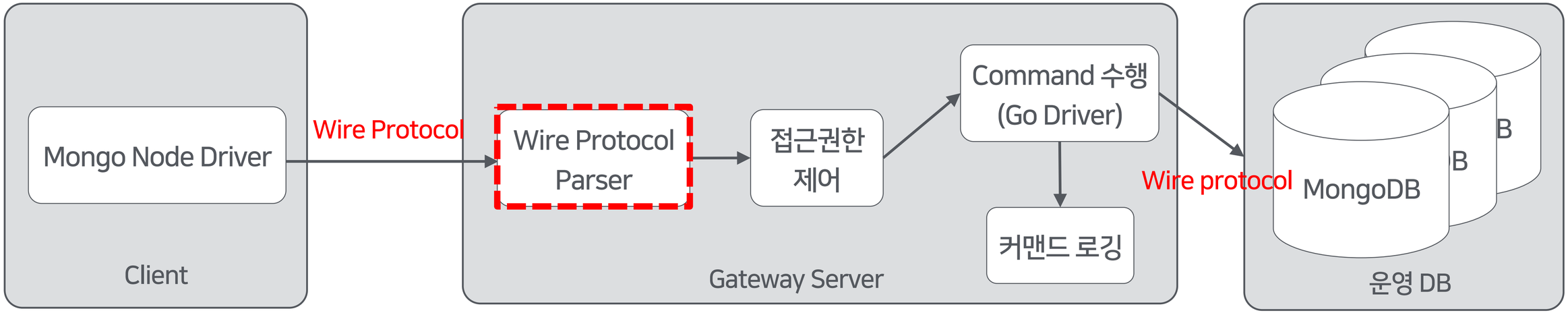
OP_MSG{MsgHeader: Header{MessageLength: 104, RequestID: 77, ResponseTo: 0, OpCode: MSG}, Flags: [], Sections: 1, Checksum: 0}, Section[0]=SingleDocument{DocumentSize: 83, Document:{find:"test",filter:{},lsid:{id:{"Subtype":4,"Data":"VqBM8gW5TY2UxDTtDIn7Iw=="}},$db:"testdb"}}

3.4.1 MongoDB Wire Protocol Parsing

DEVIEW 2021

MongoDB Driver – Server 간 통신은 Wire Protocol을 사용

- 구문 분석을 위해 Wire Protocol Parser 구현 필수



3.4.1 MongoDB Wire Protocol Parsing

DEVIEW 2021

MongoDB Wire Protocol 구조

- MongoDB Wire Protocol: socket-based request-response style protocol
- Header, Body 로 구성
- Header > OpCode 로 유형을 구분

```
struct MsgHeader {
    int32   messageLength; // total message size, including this
    int32   requestID;     // identifier for this message
    int32   responseTo;    // requestID from the original request
                           //   (used in responses from db)
    int32   opCode;        // request type - see table below for details
}
```

출처: MongoDB Wire Protocol Manual

3.4.1 MongoDB Wire Protocol Parsing

DEVIEW 2021

MongoDB Wire Protocol - OpCode

- MongoDB 3.6 이상은 OP_MSG 만 사용
- 5.0 부터는 OP_MSG 제외한 Opcode는 deprecated

Opcode Name	Value	Comment
`OP_MSG`	2013	Send a message using the format introduced in MongoDB 3.6.
`OP_REPLY` *Deprecated in MongoDB 5.0.*	1	Reply to a client request. `responseTo` is set.
`OP_UPDATE` *Deprecated in MongoDB 5.0.*	2001	Update document.
`OP_INSERT` *Deprecated in MongoDB 5.0.*	2002	Insert new document.
`RESERVED`	2003	Formerly used for OP_GET_BY_OID.
`OP_QUERY` *Deprecated in MongoDB 5.0.*	2004	Query a collection.
`OP_GET_MORE` *Deprecated in MongoDB 5.0.*	2005	Get more data from a query. See Cursors.
`OP_DELETE` *Deprecated in MongoDB 5.0.*	2006	Delete documents.
`OP_KILL_CURSORS` *Deprecated in MongoDB 5.0.*	2007	Notify database that the client has finished with the cursor.
`OP_COMPRESSED`	2012	Wraps other opcodes using compression

출처: MongoDB Wire Protocol Manual

3.4.1 MongoDB Wire Protocol Parsing

OP_MSG

- Heder: Opcode, 메시지 길이 등을 정의
- Flags: MongoDB driver -> Server 간 전달할 flag 정보
- Body: Command 정보
- Document: Bson documents
- Checksum: CRC-32C 체크섬

OP_MSG	
Header	OP_CODE, 메시지 길이 등을 정의하는 공통 헤더
Flags	exhausAllowed, moreToCome, checkSumPresent
Body	{"명령":"대상"} 형식의 command 정보. (예: {"insert":"collection"})
Document	BSON Documents (Array)
Checksum	CRC-32C checksum (Optional)

3.4.2 접근 권한 제어

접근 권한 제어 방법

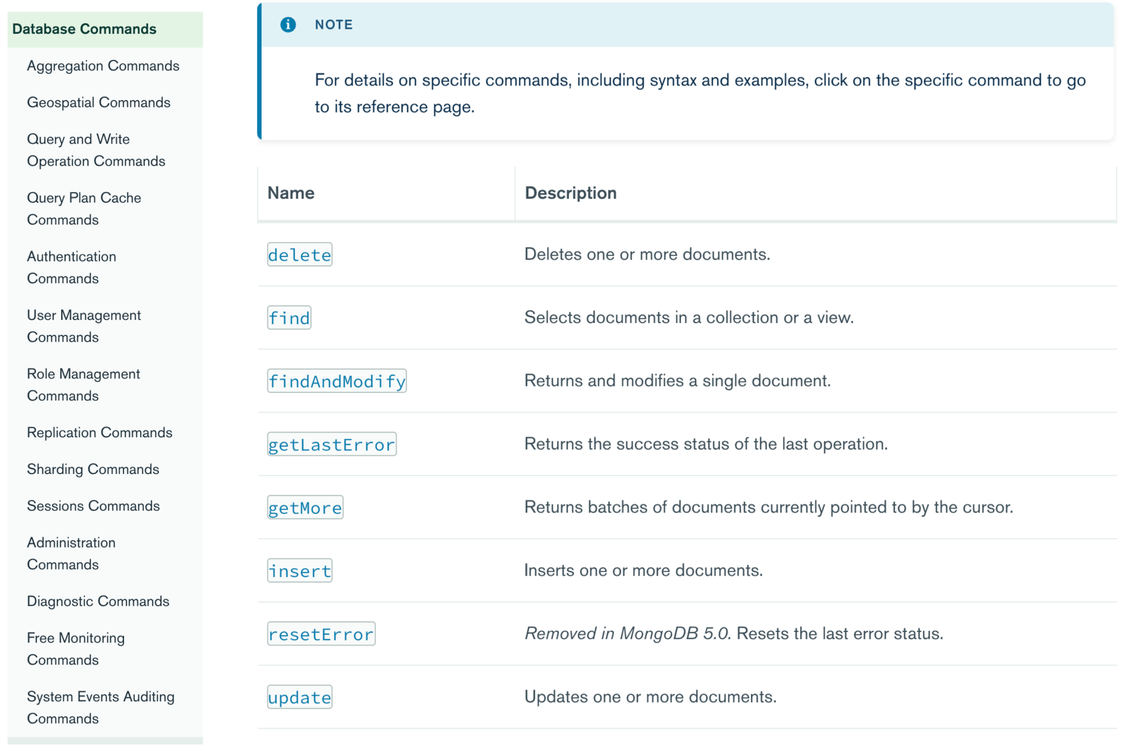
- 사용자의 권한은 SELECT, INSERT, UPDATE, DELETE, DDL 로 정의
- 모든 MongoDB Command를 분석해서 위 권한 유형에 맵핑
- 사용자가 가진 권한과 요청한 Command 유형을 비교하여 수행 제어

3.4.2 접근 권한 제어

DEVIEW 2021

권한 맵핑을 위해 모든 Command 파악 필요

- MongoDB 4.4 기준 공식 문서 146개



Database Commands

- Aggregation Commands
- Geospatial Commands
- Query and Write Operation Commands
- Query Plan Cache Commands
- Authentication Commands
- User Management Commands
- Role Management Commands
- Replication Commands
- Sharding Commands
- Sessions Commands
- Administration Commands
- Diagnostic Commands
- Free Monitoring Commands
- System Events Auditing Commands

Query and Write Operation Commands

NOTE

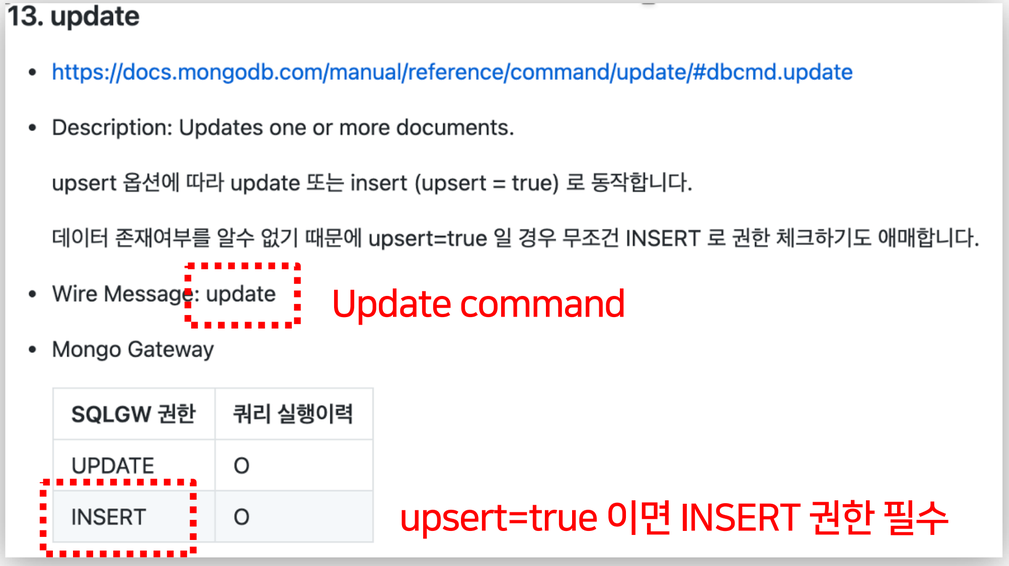
For details on specific commands, including syntax and examples, click on the specific command to go to its reference page.

Name	Description
`delete`	Deletes one or more documents.
`find`	Selects documents in a collection or a view.
`findAndModify`	Returns and modifies a single document.
`getLastError`	Returns the success status of the last operation.
`getMore`	Returns batches of documents currently pointed to by the cursor.
`insert`	Inserts one or more documents.
`resetError`	*Removed in MongoDB 5.0.* Resets the last error status.
`update`	Updates one or more documents.

3.4.2 접근 권한 제어

MongoDB Command 권한 맵핑

- SQL처럼 CRUD로 1:1 맵핑이 어려움
  (CRUD=SELECT, INSERT, UPDATE, DELETE)
- Update command 의 경우 INSERT, UPDATE 두개 권한을 필수로 정의



13. update

- https://docs.mongodb.com/manual/reference/command/update/#dbcmd.update
- Description: Updates one or more documents.

  upsert 옵션에 따라 update 또는 insert (upsert = true) 로 동작합니다.

  데이터 존재여부를 알수 없기 때문에 upsert=true 일 경우 무조건 INSERT 로 권한 체크하기도 애매합니다.
- Wire Message: update
- Mongo Gateway

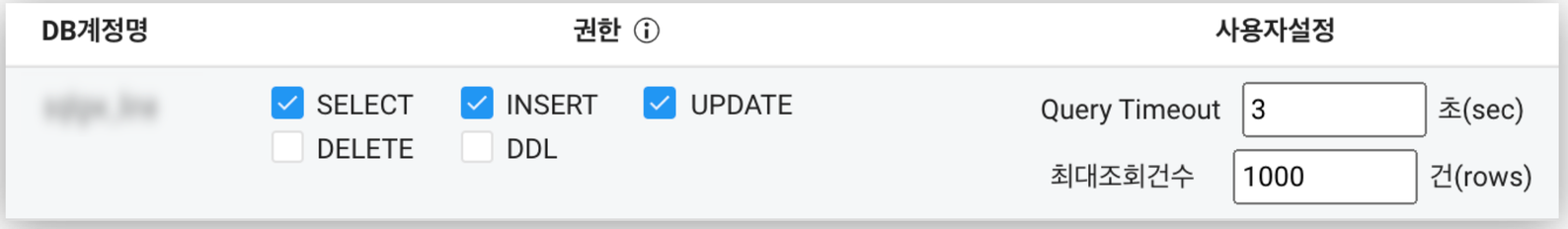
SQLGW 권한	쿼리 실행이력
UPDATE	O
INSERT	O

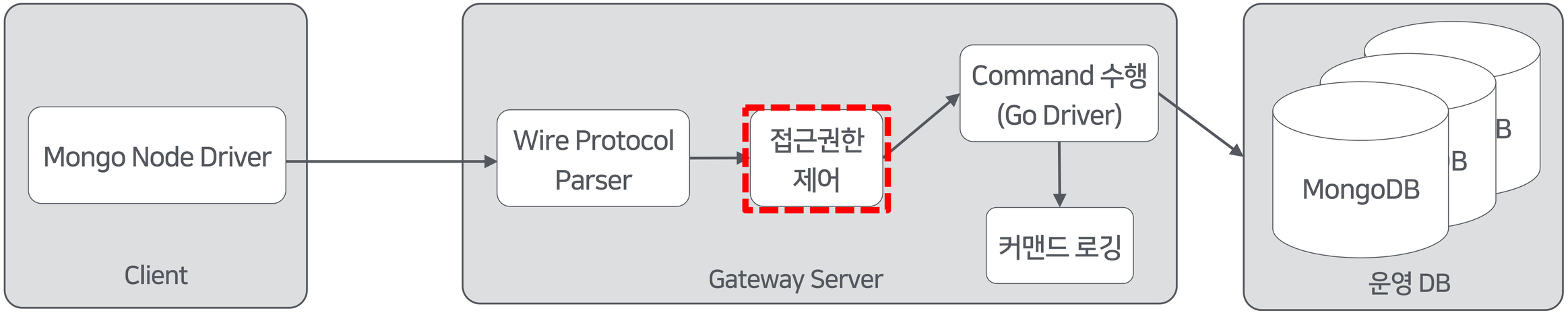
Command 권한 분류 정리 中

3.4.2 접근 권한 제어

사용자는 MongoDB 계정의 권한과 설정을 신청해서 사용

사용자
권한 신청

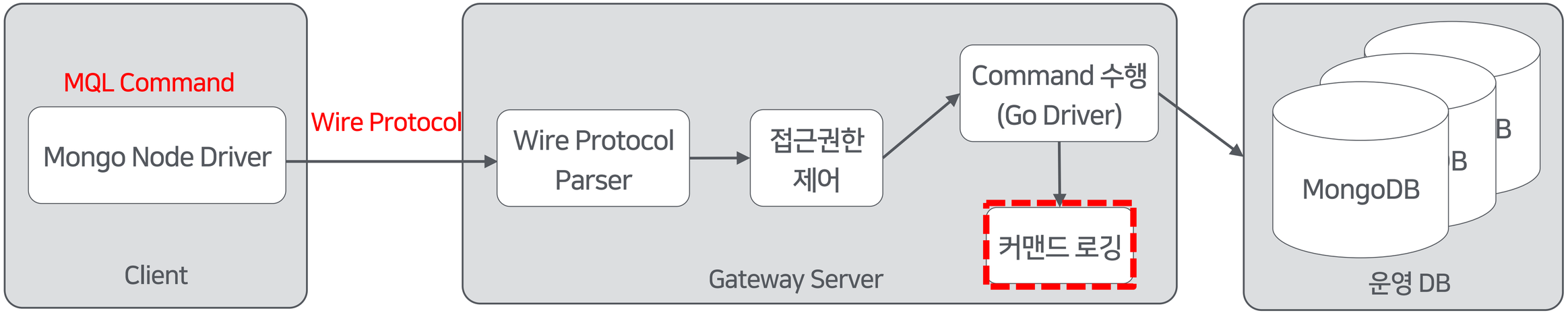




3.4.3 Command 로깅

사용자가 실행한 Command 를 그대로 로깅하기 어려움

- Client 에서 입력 가능한 Command 유형이 다양함
- Wire Protocol 을 통해 서버로 전달 되면서 원본 Command를 알수 없음
- 공통 포멧으로 로그 기록 필요



3.4.3 Command 로깅

runCommand method

- 사용자가 입력한 다양한 형태의 Command 를 공통 포멧으로 기록 필요
- MongoDB 는 db.runCommand({}) 문법으로 모든 쿼리 표현 가능

```
db.test.find( { "name.last": "Hopper" } )
```

원본 Command

```
db.runCommand( { find: "test", filter: { "name.last": "Hopper" } })
```

변환된 runCommand

3.4.3 Command 로깅

DB switching

- Mongo Shell은 use <database> 명령어로 DB 를 변경
- Command 실행 시 db 명이 생략됨
- db.collection.find() Command는 어떤 DB 에 Command를 실행했는지 알 수 없음
- db.getSiblingDB("DB").runCommand({}) 로 Command 형식 통일 가능

```
use testdb
db.test.find( { "name.last": "Hopper" } )
```

```
db.getSiblingDB("testdb").runCommand( { find: "test", filter: { "name.last": "Hopper" } })
```

3.5.1 그 외 처리한 사항 - Kill Command

DEVIEW 2021

Kill Command 실행

1. 실행중인 Command 조회
   db.currentOp()

2. Kill Command
   db.adminCommand({ "killOp": 1, "op": <operationID> })

3.5.1 그 외 처리한 사항 - Kill Command

DEVIEW 2021

SQL Gateway 에서 실행한 Command 만 Kill 해야 함

1. 접속 옵션 AppName 에 AppName+SessionID 를 저장
2. Kill 대상 조회 시 결과 필터링 currentOp > inprog

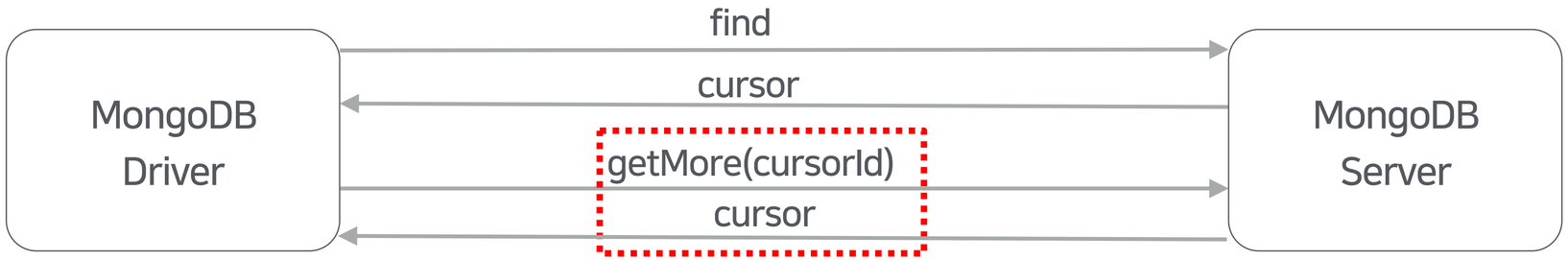

```
"inprog" : [
        {
                "type" : "op",
                "host" : "dbip-mongotest-001",
                "desc" : "conn930033",
                "connectionId" : 930033,
                "clientMetadata" : {
                        "driver" : {
                                "name" : "mongo-go-driver",
                                "version" : "v1.3.4"
                        },
                        "os" : {
                                "type" : "darwin",
                                "architecture" : "amd64"
                        },
                        "platform" : "go1.15.7",
                        "application" : {
                                "name" : "mongo-gateway_3-e59be1c0-cb6d-4772-ab0f-c4b8322183c6"
                        }
                },
                "active" : true,
                "currentOpTime" : "2021-05-11T12:12:50.906+09:00",
        }
],
"ok" : 1
```


App Name + SessionID

3.5.2 그 외 처리한 사항 - Cursor Fetch

MongoDB 의 Cursor Fetch 방식

- find, aggregate 는 cursor 를 리턴
- 다음 데이터를 가져올때 getMore command 를 사용
- getMore 는 cursor 가 열린 서버로 요청 해야 함

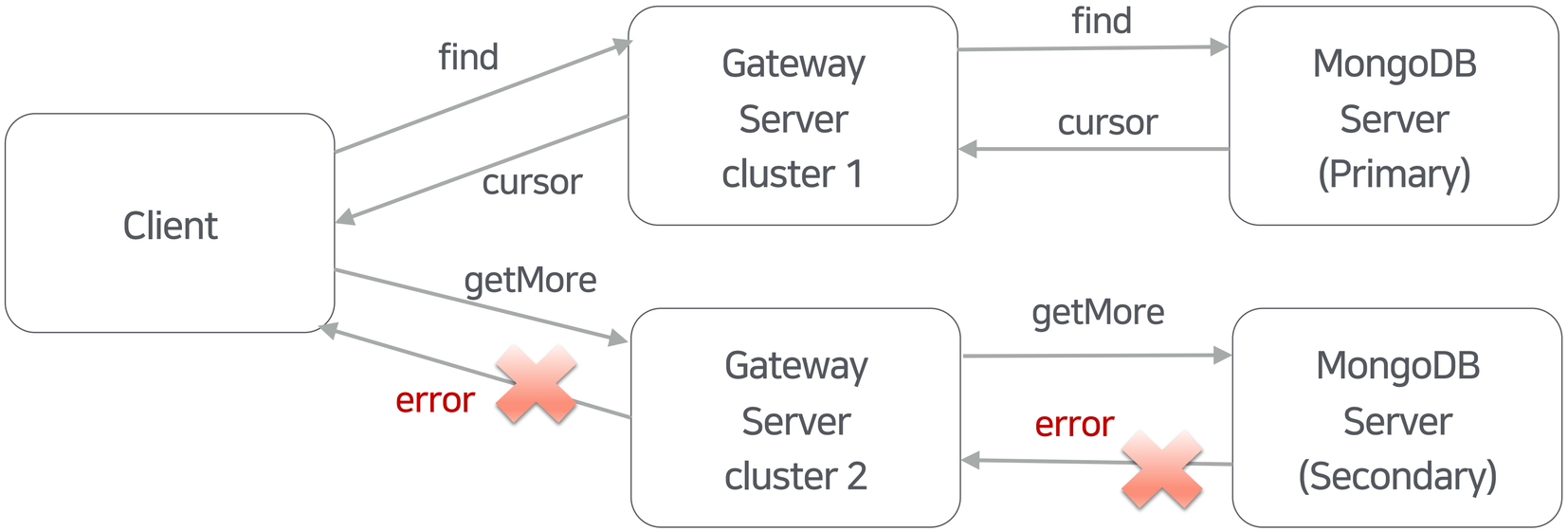


3.5.2 그 외 처리한 사항 - Cursor Fetch

DEVIEW 2021

Gateway Server 를 통하는 경우 발생했던 이슈

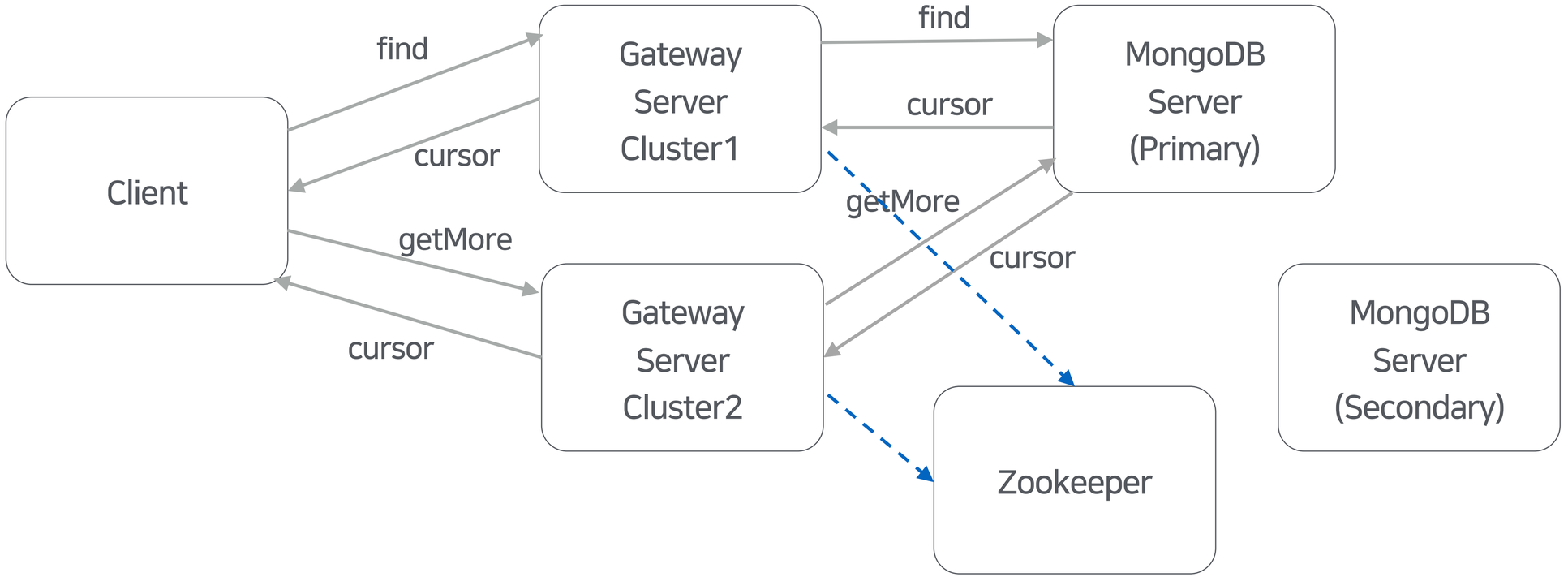
- Driver 에서 Connection Pool 로 동시에 요청될 수 있음
- getMore 요청이 다른 서버로 요청될 수 있음
- MongoDB 에 Cursor session 이 없는 경우 에러 발생

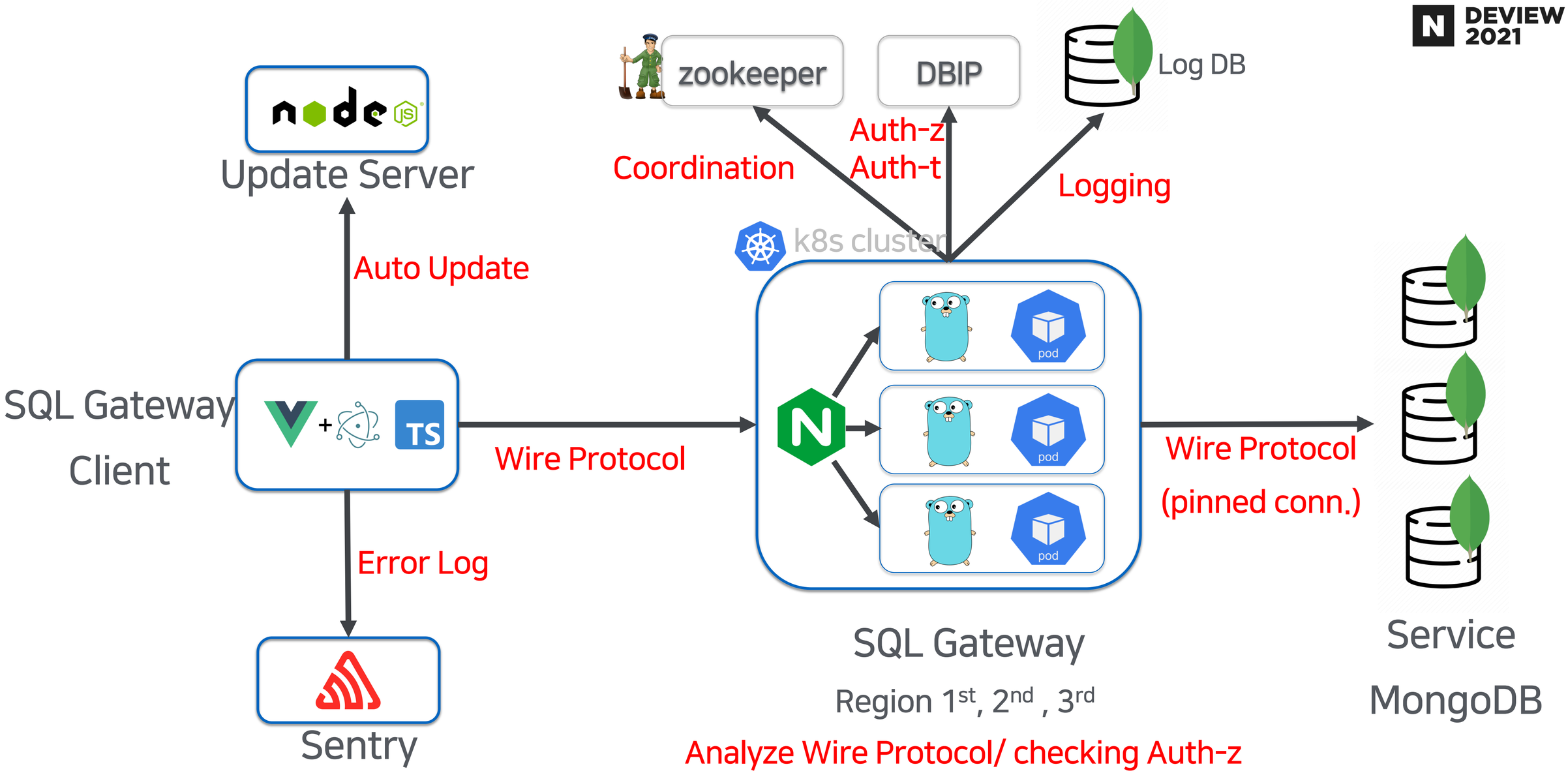


3.5.2 그 외 처리한 사항 - Cursor Fetch

zookeeper 활용한 sticky session 구현

- 접속 서버 정보를 zookeeper 에 저장
- 동일 세션인 경우 이전 접속 서버로 요청





SQL Gateway for MongoDB Overall Architecture

4.Lessons Learned

4.1 Lessons Learned

MongoDB 접근제어 시스템 구축간 중점사항

- 모든 Command를 분석하고 명확하게 권한을 맵핑해야만 보안 이슈가 생기지 않음
- 함수화 가 가능한 Command 특성을 반드시 고려
- 많은 사용자가 몰리거나 대용량 처리 등 부하를 고려하여 Auto Scale 가능한 구조 (k8s cluster 등) 로 설계

4.1 Lessons Learned

Desktop Application 구축 고려사항

- 패키징, 다양한 OS지원, 자동업데이트, 에러 트래킹, Signing 등 할 작업이 많음
- Electron적용 고려시 장단점을 명확히 파악해서 선택함이 필요
- 보안측면을 고려해서 반드시 application 위변조 방지 기능 추가

Contributor

임근대 (mordern.lim@navercorp.com)

김도영 (kimdo489@nts-corp.com)

윤성현 (sunghyun.yun@nts-corp.com)

지화영 (hwayeongji@nts-corp.com)

Thank You