

# 네이버 로그를 지탱하는 힘 (DataStore 로그 저장소)

강민우 / 이윤경

# CONTENTS

DEVIEW  
2019

## 1장. 로그 저장 이야기

로그 흐름의 문제

로그 파이프라인

로그 저장소의 주요 기능

## 2장. 로그 활용 이야기

네이버 검색 로그 활용

SQL Interface

로그 분석

# 1. 로그 저장 이야기

# 기존 문제

## 생산자와 소비자의 강한 결합

이 로그는 어디서 구하지?



소비자

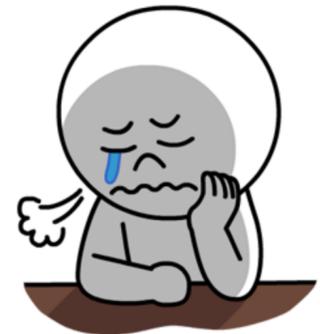
# 기존 문제

## 생산자와 소비자의 강한 결합



생산자

서버를 열어줘야 할까?  
내가 데이터를 넣어줘야 할까?  
어떻게 해야 할까?



소비자

# 기존 문제

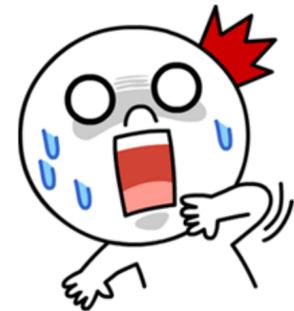
## 생산자와 소비자의 강한 결합



생산자

수십, 수백대 서버에서 남는 로그를  
어떻게 잘 가지고 오지?

서버를 열어줘야 할까?  
내가 데이터를 넣어줘야 할까?  
어떻게 관리할까?



소비자

# 기존 문제

## 생산자와 소비자의 강한 결합



생산자

이 필드가 무슨 의미인지 모르겠네...

수십, 수백대 서버에서 남는 로그를  
어떻게 잘 가지고 오지?

서버를 열어줘야 할까?  
내가 데이터를 넣어줘야 할까?  
어떻게 해야 할까?



소비자

# 기존 문제

## 생산자와 소비자의 강한 결합



생산자

이 필드가 무슨 의미인지 모르겠네...

매번 반복적인 커뮤니케이션..  
없앨 수는 없을까?

이렇게 날마다 오시?

서버를 열어줘야 할까?  
내가 데이터를 넣어줘야 할까?

구하지?



소비자

# 기존 문제

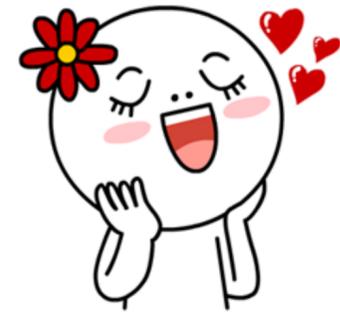
## 생산자와 소비자의 분리



생산자

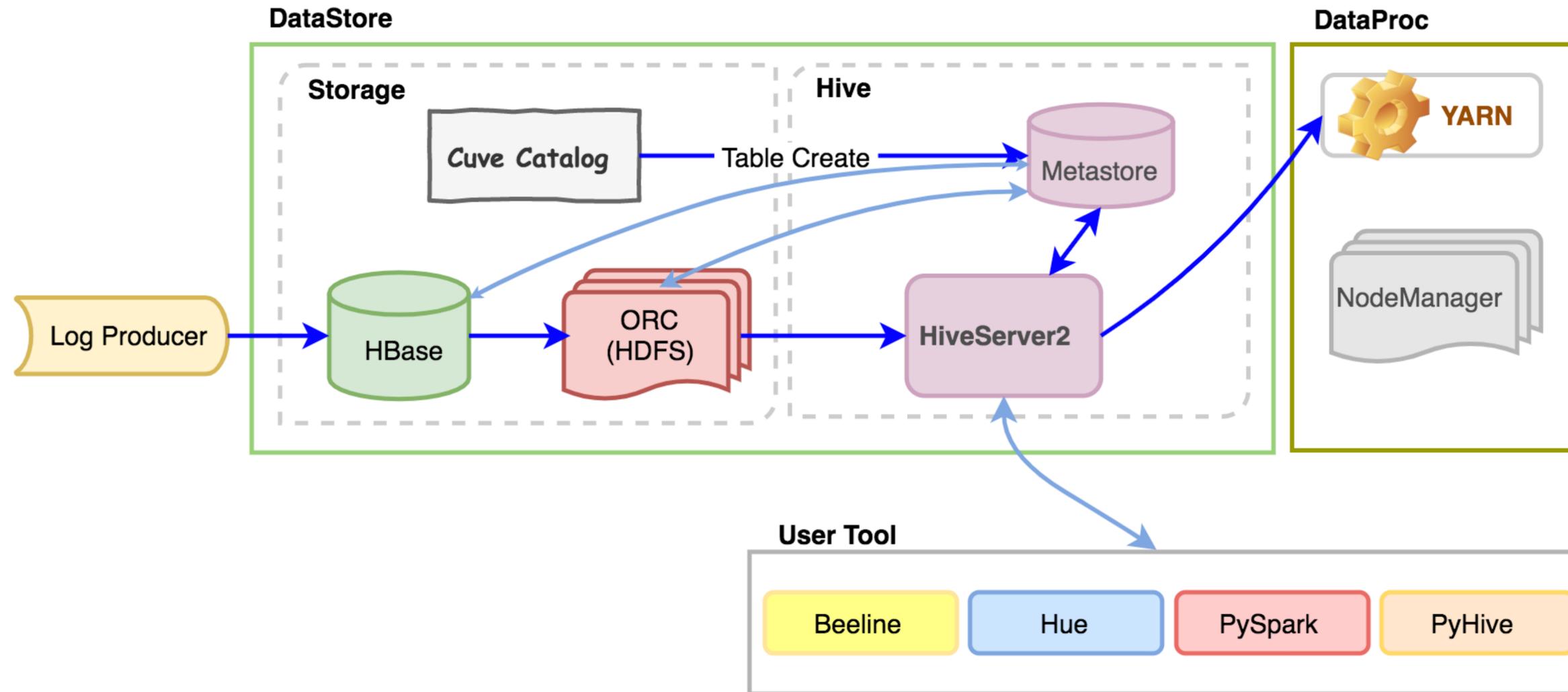


DataStore 로그 저장소

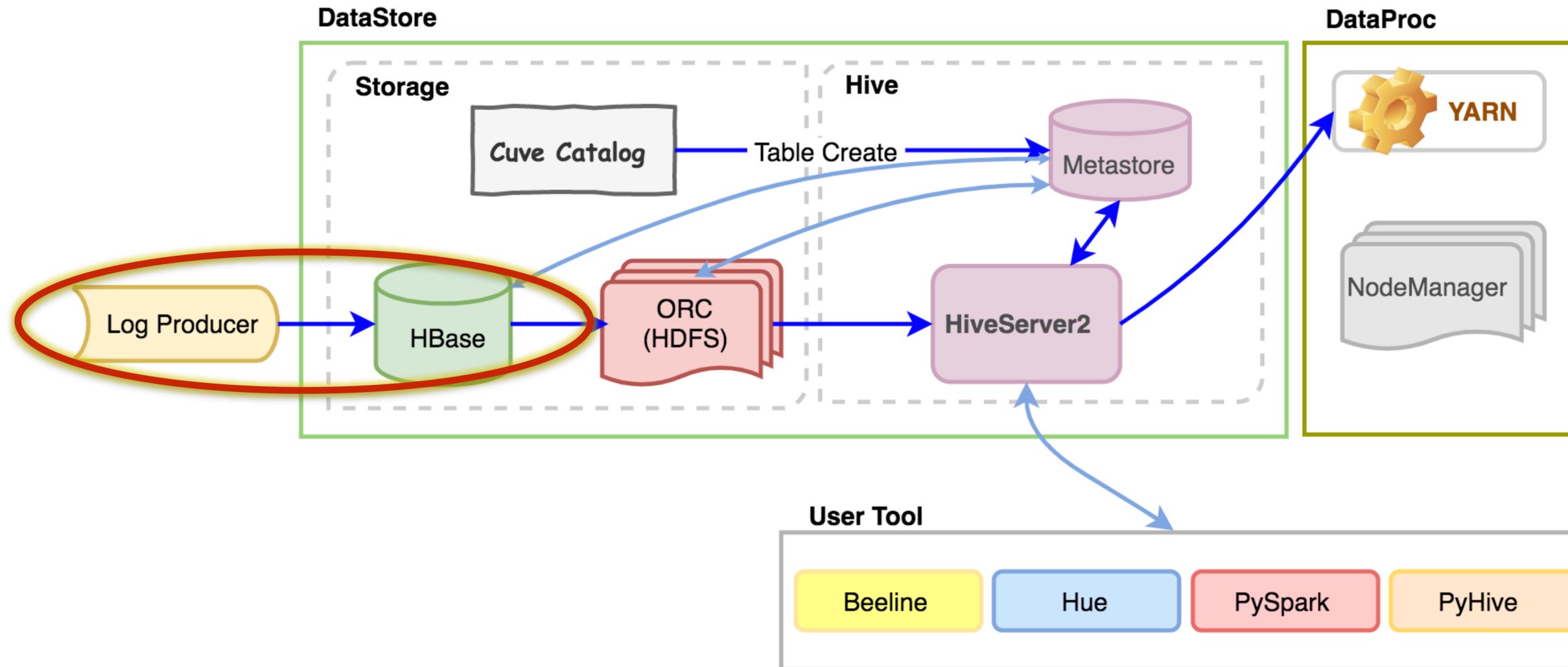


소비자

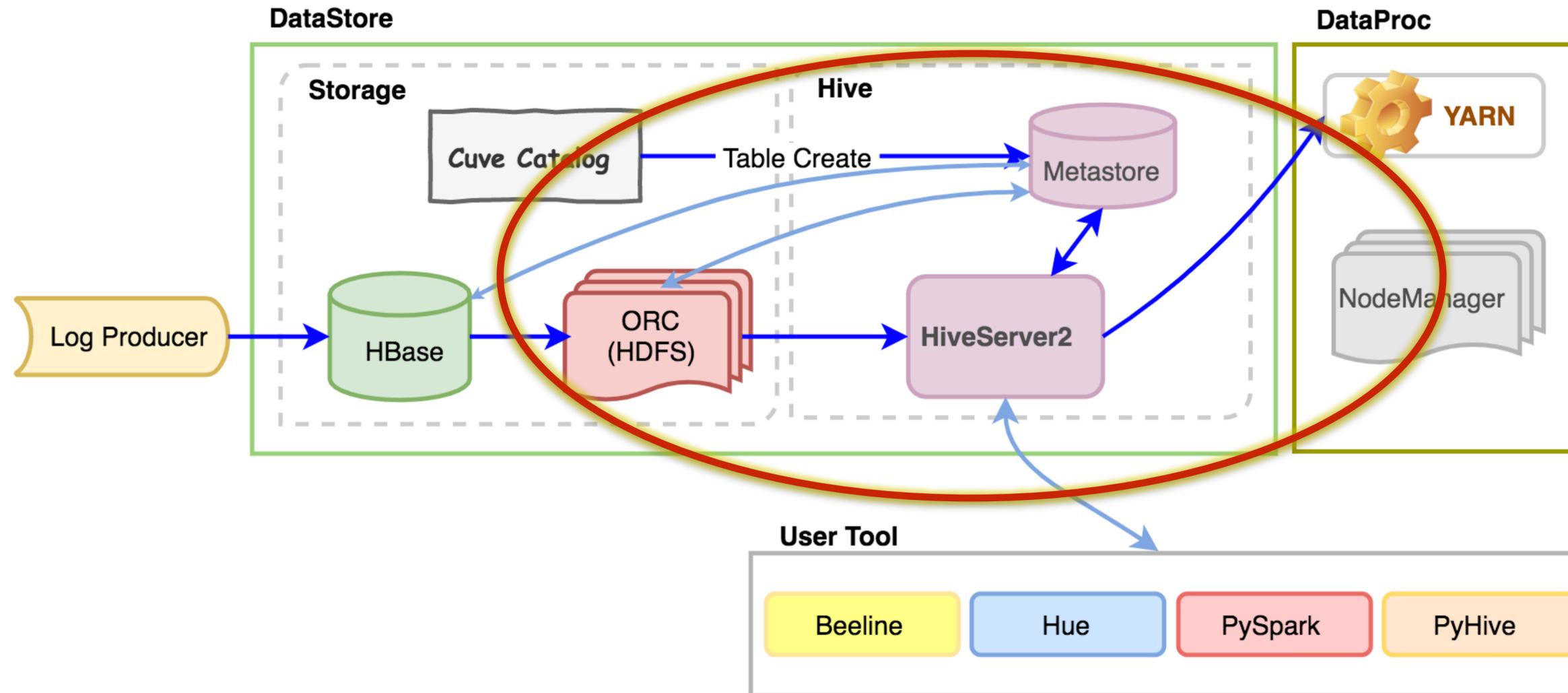
# 로그 파이프라인



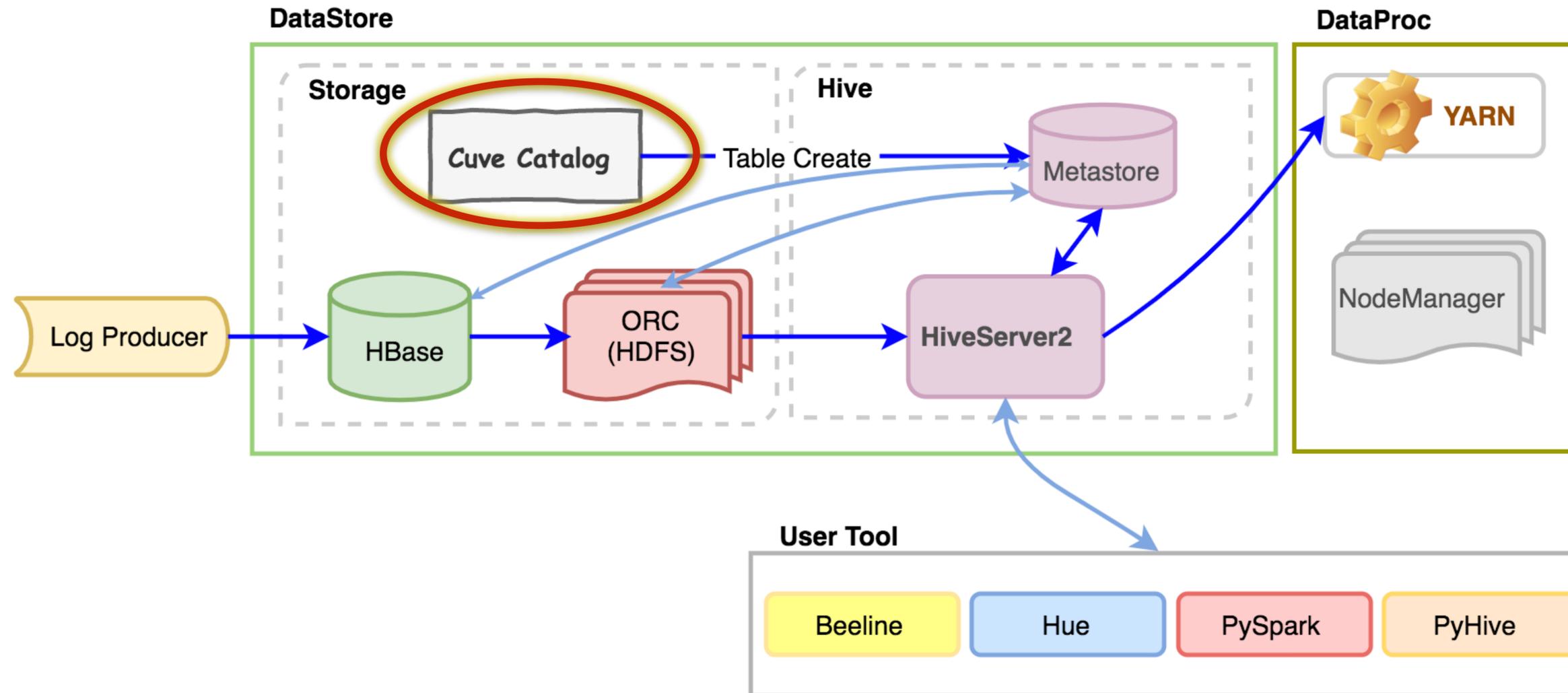
# 로그 파이프라인



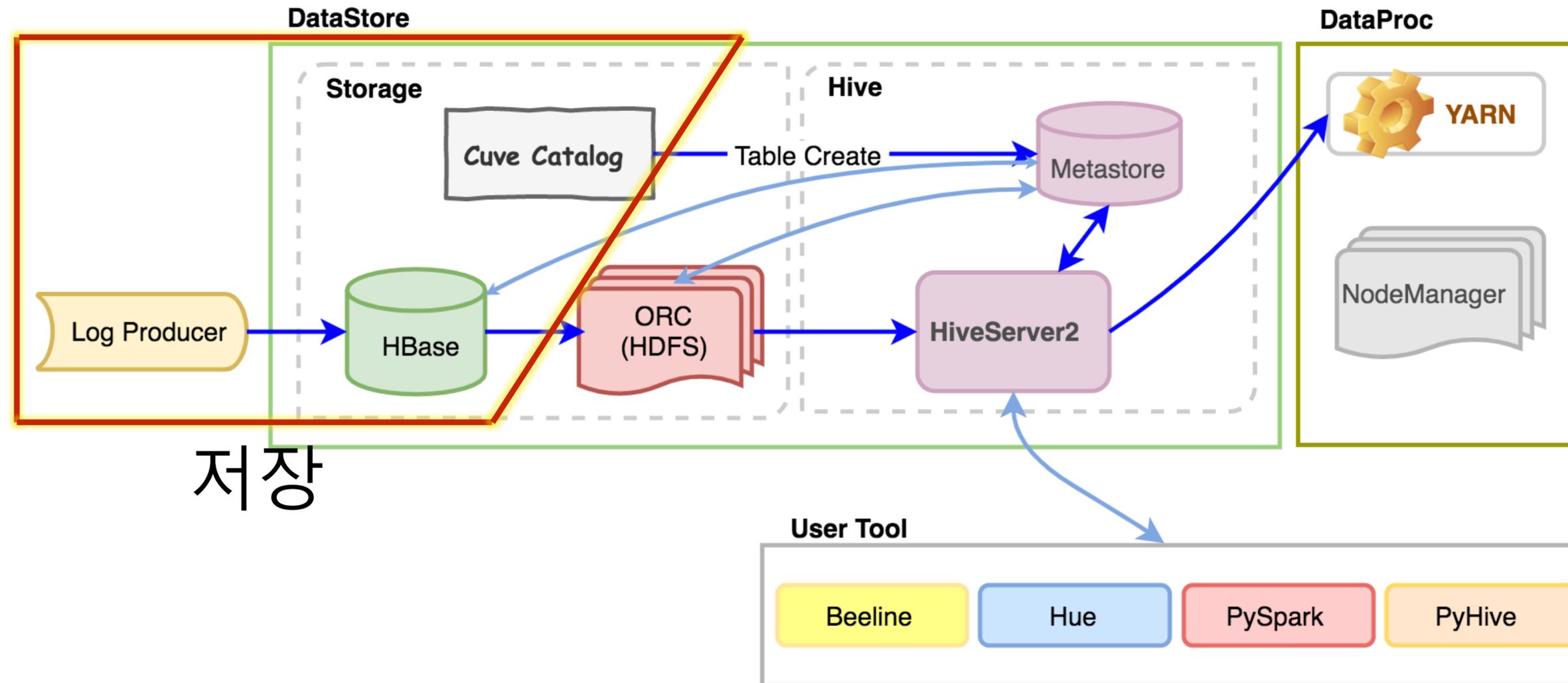
# 로그 파이프라인



# 로그 파이프라인



# 로그 파이프라인



# 주요 기능



**유실없는** 로그 저장,  
내가 **원하는기간**에 생성된 로그를 **필요한필드**만  
**동일한인터페이스**를 통해 받아 갈 수 있는 플랫폼

그외에도,

- 보안 정보에 대한 처리
- 사용자 관리

# 주요 기능

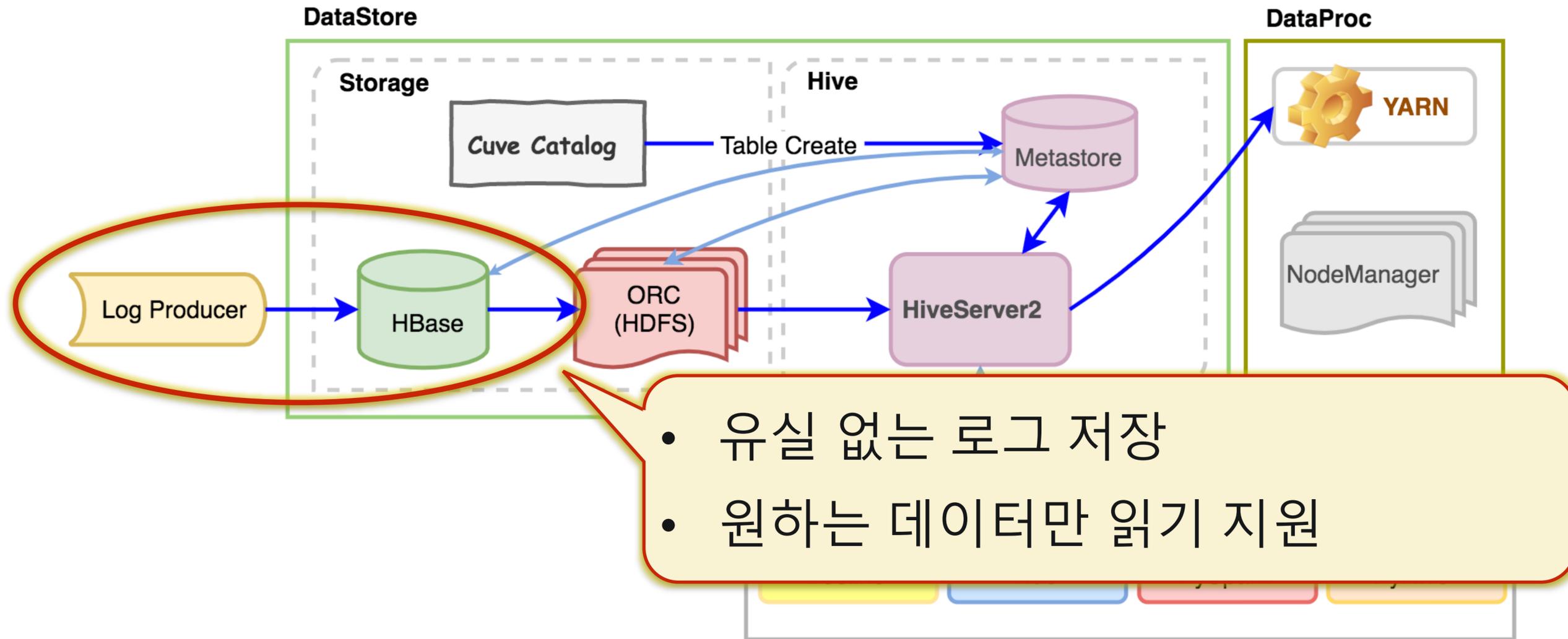
**유실없는** 로그 저장,  
내가 **원하는기간**에 생  
**동일한인터페이스**를

보안 정보 분리 저장, 주기적 삭제, 데이터 접근 제어를  
통해 안전한 데이터 활용 지원

그외에도,

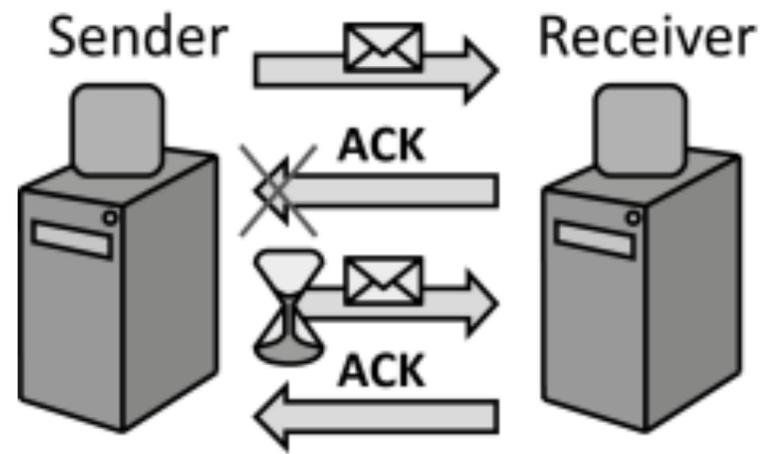
- 보안 정보에 대한 처리
- 사용자 관리

# 로그 파이프라인



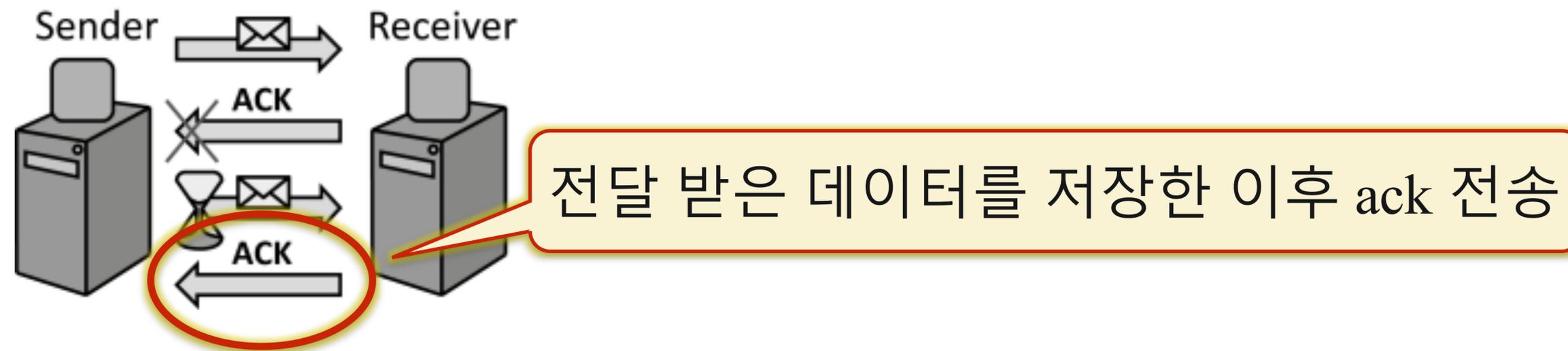
# 유실 없는 로그 저장

## At-least-once Delivery



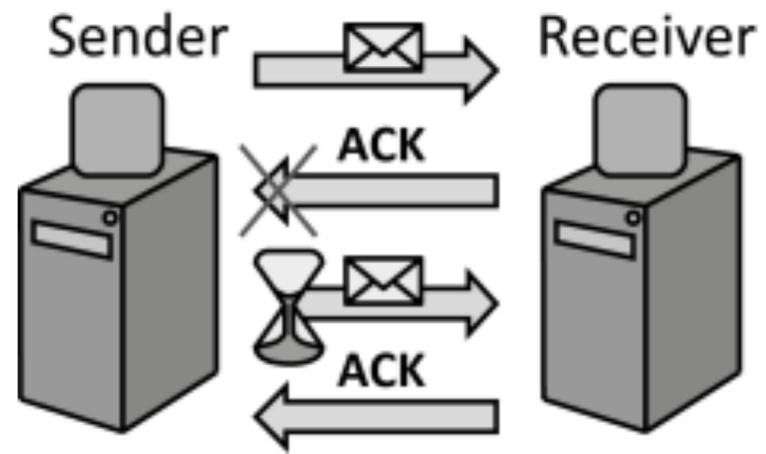
# 유실 없는 로그 저장

## At-least-once Delivery



# 유실 없는 로그 저장

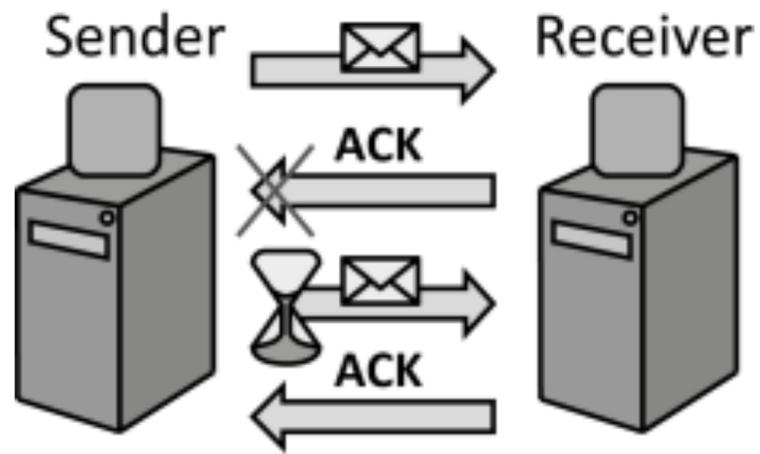
## At-least-once Delivery



데이터 중복이 발생 할 수 있음

# 유실 없는 로그 저장

## At-least-once Delivery



데이터 중복이 발생 할 수 있음

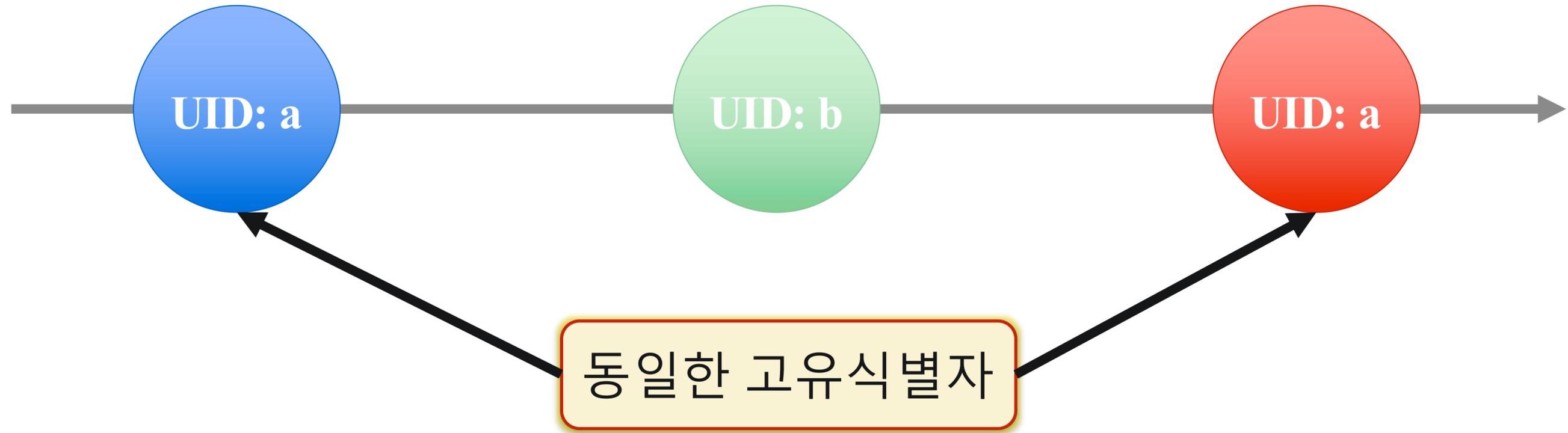
해결 방법: 외부 저장소의 고유 식별자를 사용한 idempotent 쓰기

# 중복 저장 제거

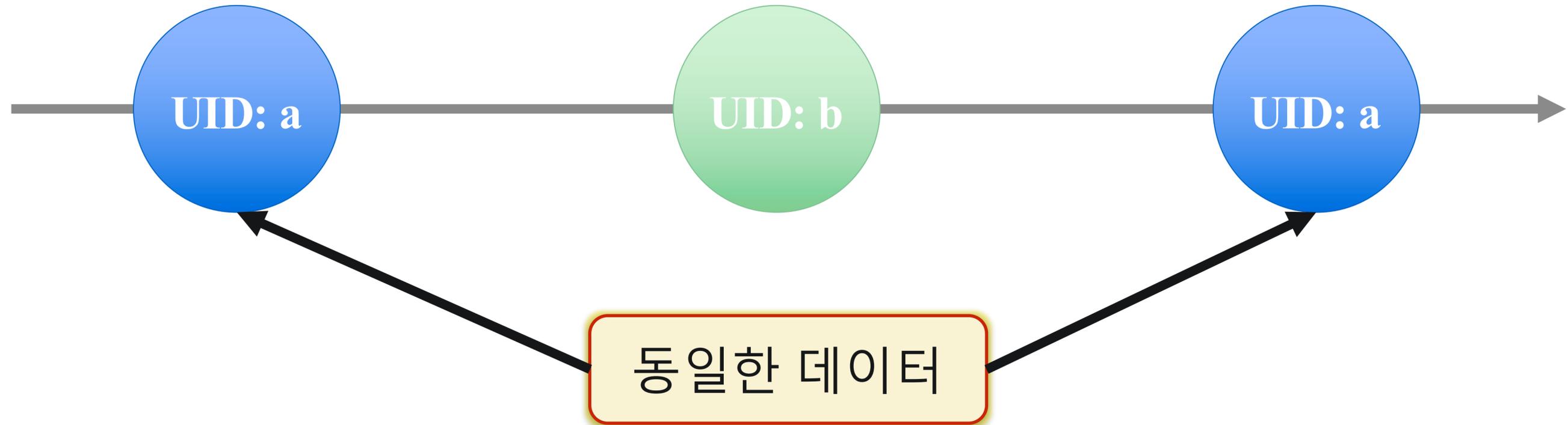
## At-least-once Delivery



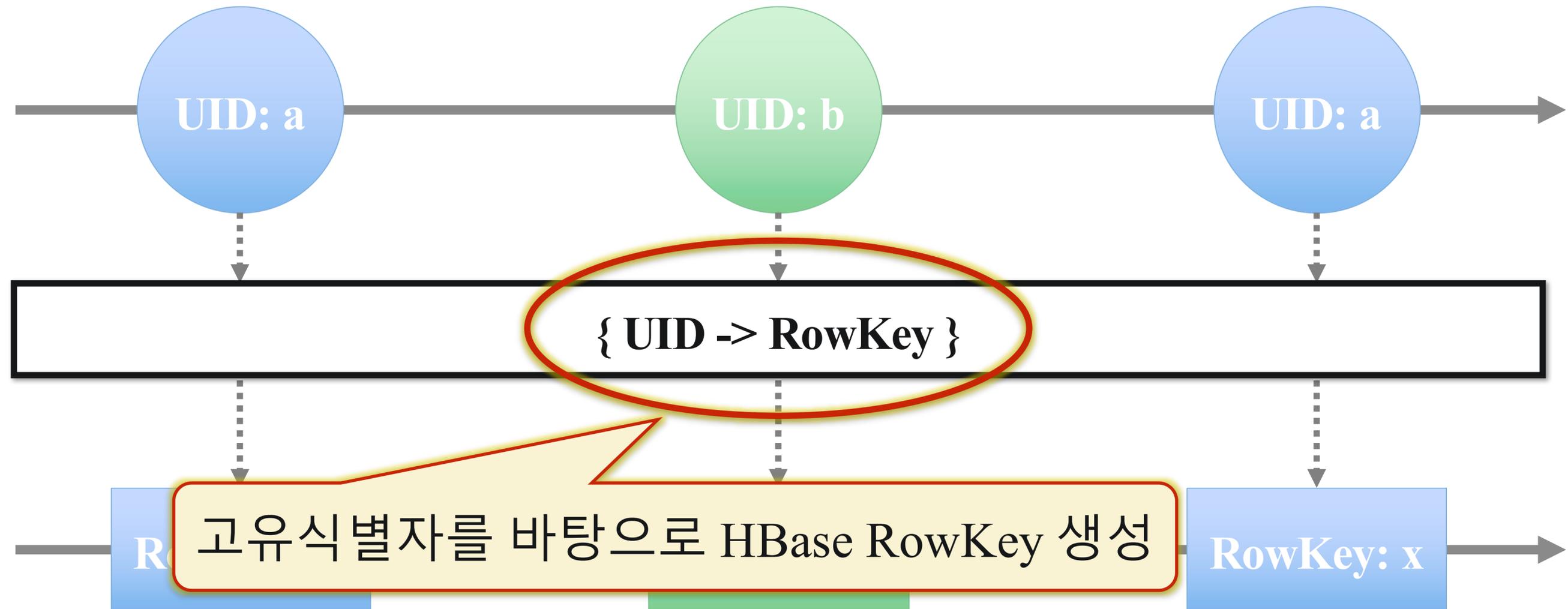
# 중복 저장 제거



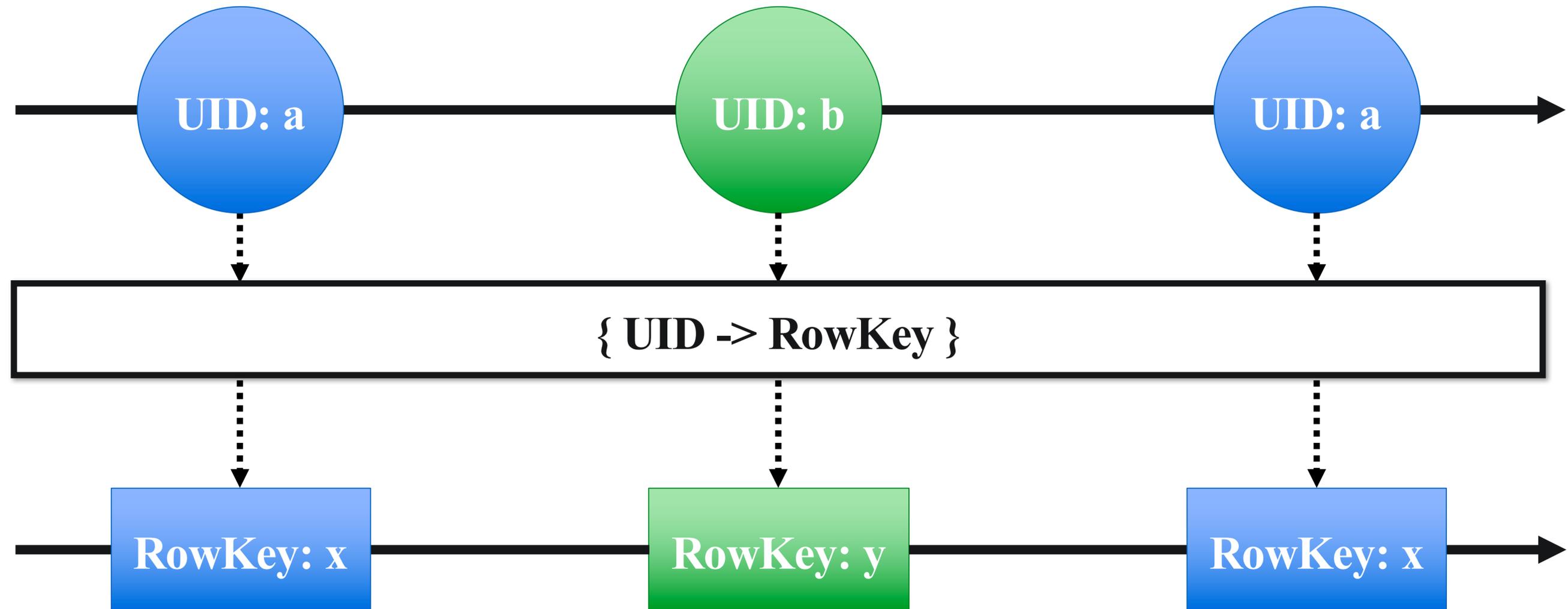
# 중복 저장 제거



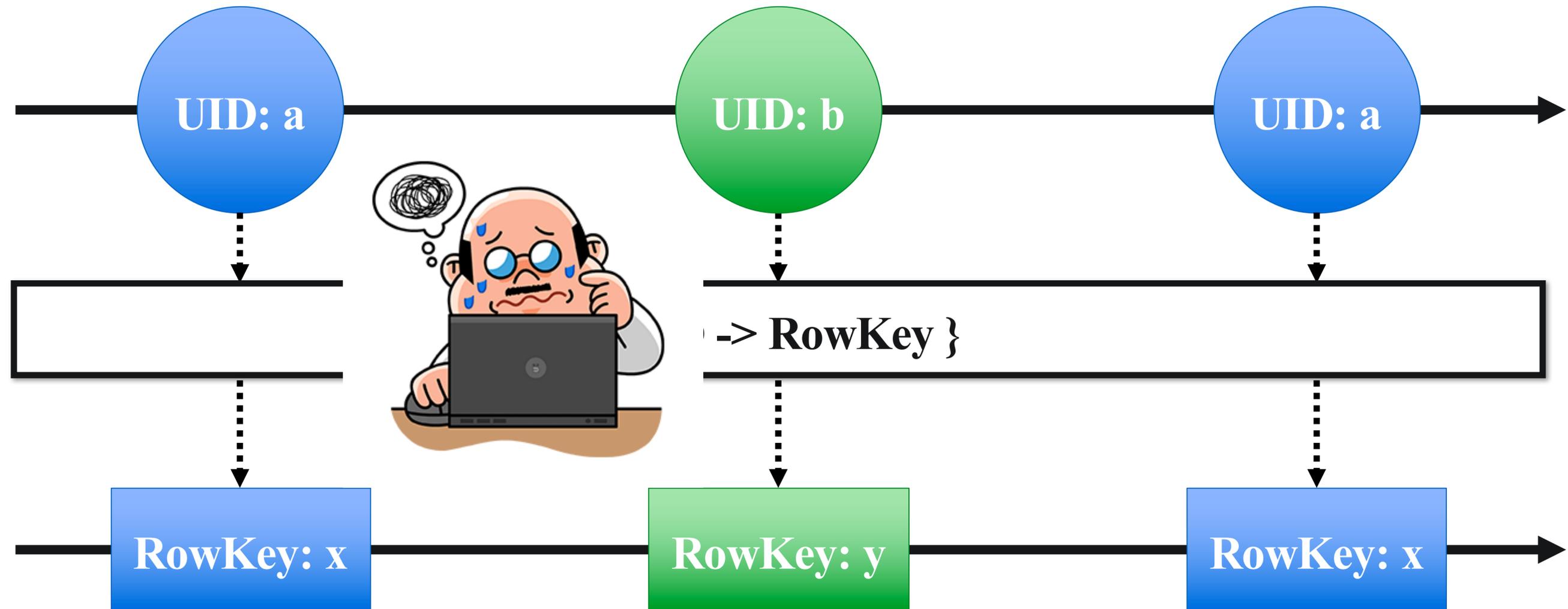
# 중복 저장 제거



# 중복 저장 제거



# 중복 저장 제거



# 고유 식별자 할당하기



## 1) Hash 값

"GET /search.naver?query=데뷰 HTTP/1.1" 200

"GET /search.naver?query=데뷰 HTTP/1.1" 200

"GET /search.naver?query=데뷰 HTTP/1.1" 200

"GET /search.naver?query=데뷰 HTTP/1.1" 200



내용만 동일한 로그

# 고유 식별자 할당하기



## 1) Hash 값

"GET /search.naver?query=데뷰 HTTP/1.1" 200

"GET /search.naver?query=데뷰 HTTP/1.1" 200

"GET /search.naver?query=데뷰 HTTP/1.1" 200

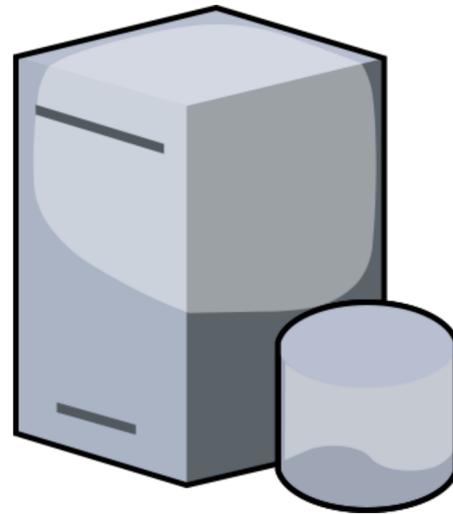
"GET /search.naver?query=데뷰 HTTP/1.1" 200



동일한 Hash 값

# 고유 식별자 할당하기

## 2) 시퀀스 값 발급



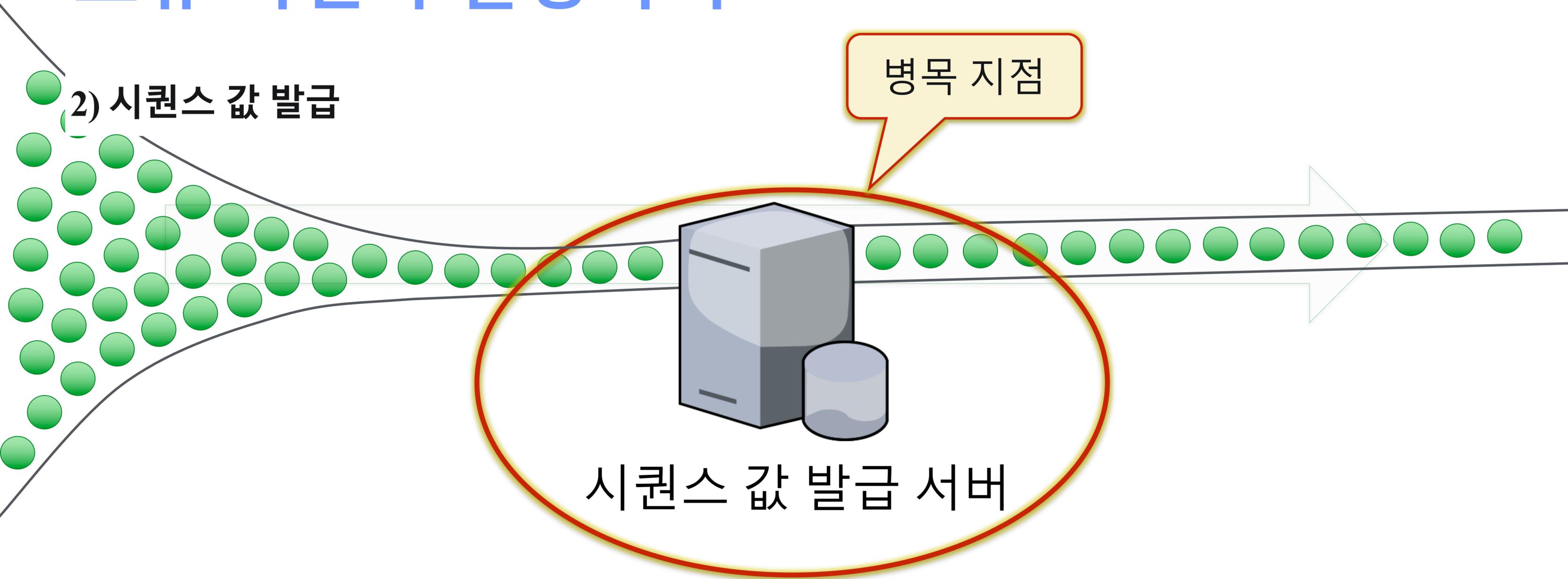
시퀀스 값 발급 서버

# 고유 식별자 할당하기

2) 시퀀스 값 발급

병목 지점

시퀀스 값 발급 서버



# 고유 식별자 할당하기



## 3) UUID (범용 고유 식별자)

범용 고유 식별자는 소프트웨어 구축에 쓰이는 식별자 표준으로, 개방 소프트웨어 재단(OSF)이 분산 컴퓨팅 환경(DCE)의 일부로 표준화하였다.

# 고유 식별자 할당하기



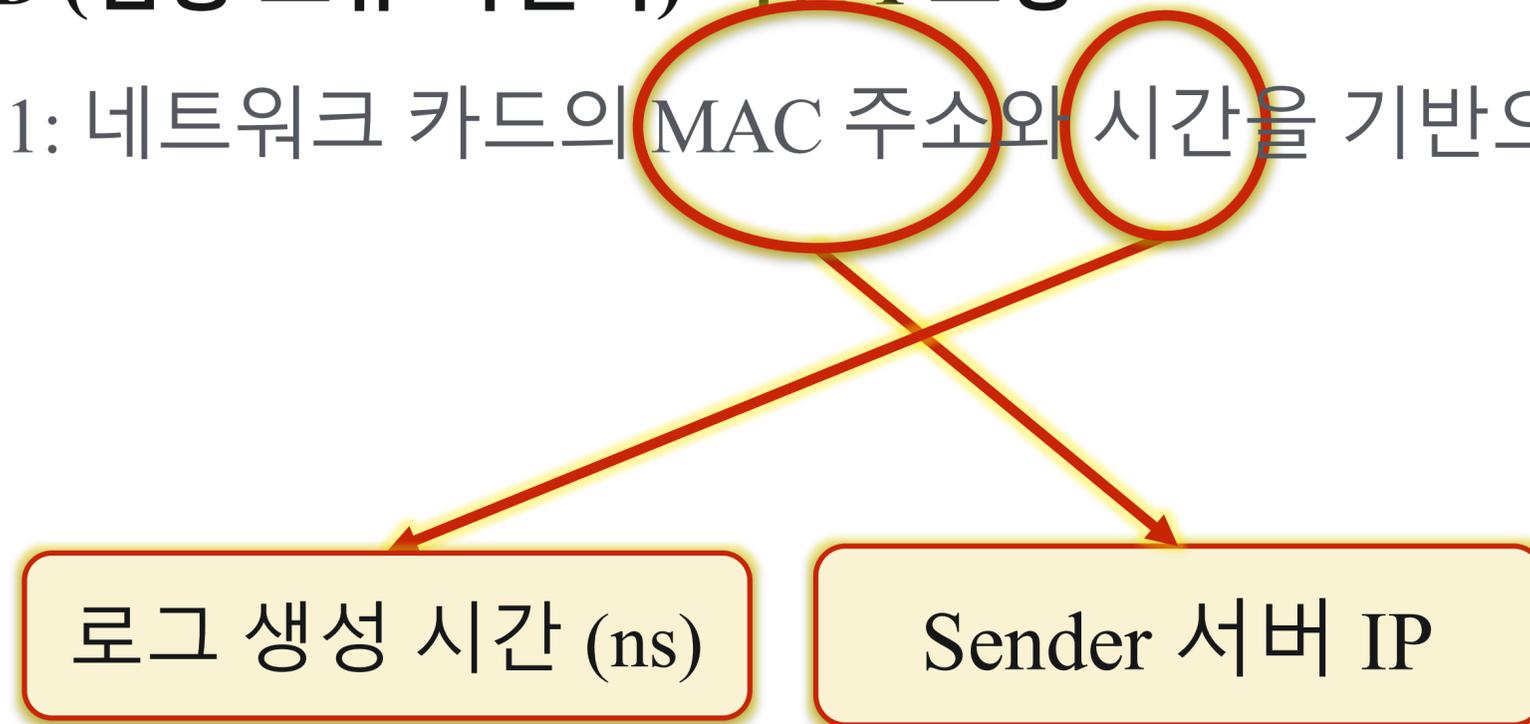
## 3) UUID (범용 고유 식별자) 버전 1 모방

버전1: 네트워크 카드의 MAC 주소와 시간을 기반으로 유니크한 ID 생성

# 고유 식별자 할당하기

## 3) UUID (범용 고유 식별자) 버전 1 모방

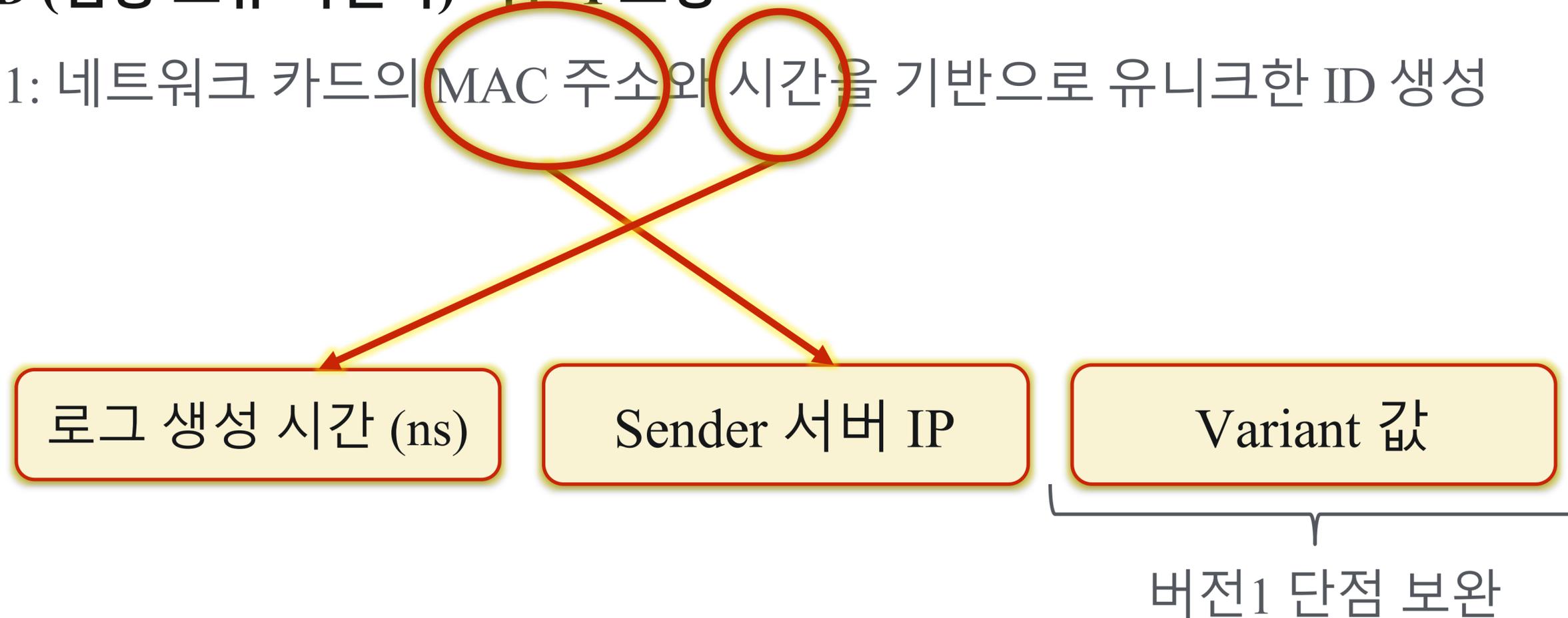
버전1: 네트워크 카드의 MAC 주소와 시간을 기반으로 유니크한 ID 생성



# 고유 식별자 할당하기

## 3) UUID (범용 고유 식별자) 버전 1 모방

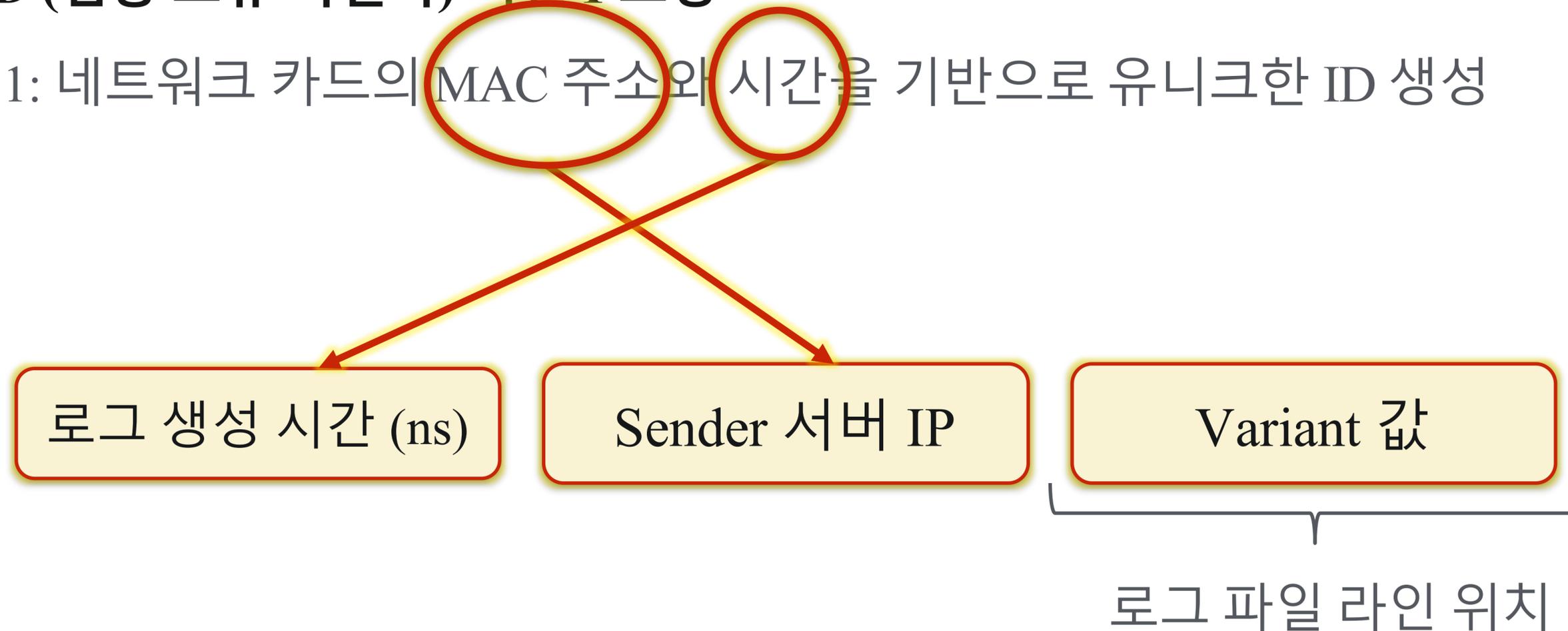
버전1: 네트워크 카드의 MAC 주소와 시간을 기반으로 유니크한 ID 생성



# 고유 식별자 할당하기

## 3) UUID (범용 고유 식별자) 버전 1 모방

버전1: 네트워크 카드의 MAC 주소와 시간을 기반으로 유니크한 ID 생성



# 고유 식별자 할당하기

## 3) UUID (범용 고유 식별자) 버전 1 모방

버전1: 네트워크 카드의 MAC 주소와 시간을 기반으로 Unique Key 생성

로그 생성 시간 (ns)

Sender 서버 IP

Variant 값

# RowKey 디자인



Category 1

Category 2

Category 3

# RowKey 디자인



Region

Group

MessageType

# RowKey 디자인



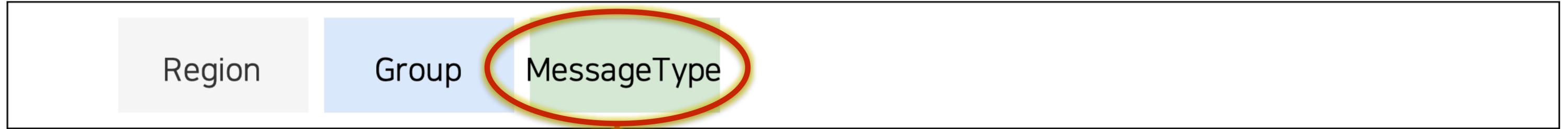
**Region:** 데이터가 생산되는 지역을 구분하기 위한 최상위 카테고리

# RowKey 디자인



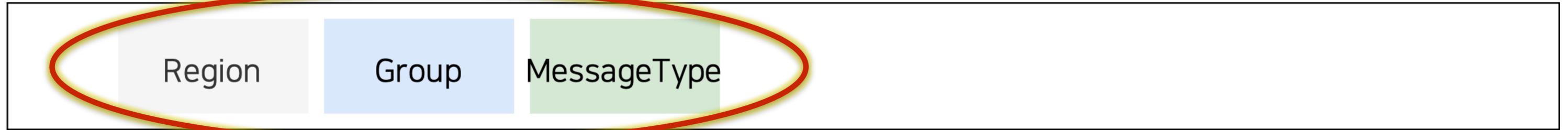
**Group:** Region 안에서 서비스를 구분하기 위한 카테고리

# RowKey 디자인



**MessageType:** 서비스내에서 데이터를 구분하기 위한 카테고리

# RowKey 디자인



Region, Group, MessageType을 줄여서



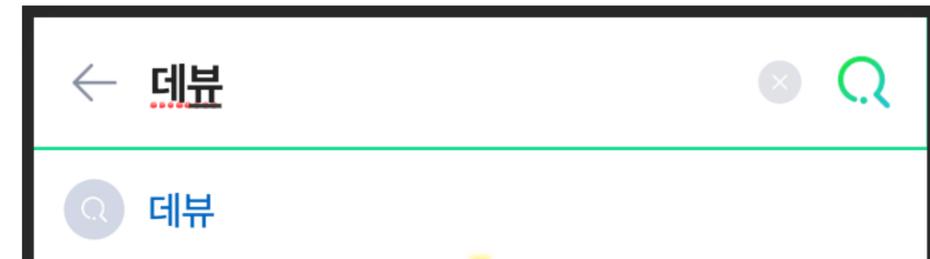
# RowKey 디자인



R korea

G naver\_search

M query\_log



"GET /search.naver?query=데뷰 HTTP/1.1" 200

# RowKey 디자인



앞에서 고유 식별자 할당을 위해서 만든 **UniqueKey**

# 원하는 기간 데이터 연속 읽기



## HBase Table

R	korea	G	naver_search	M	query_log	Time	15713377 <b>70000</b> ...
R	korea	G	naver_search	M	query_log	Time	15713377 <b>80000</b> ...
R	korea	G	naver_search	M	query_log	Time	15713377 <b>90000</b> ...

시간 순서대로 연속 읽기 가능

# Hot Spot



순차적으로 증가하는 값

# Hot Spot



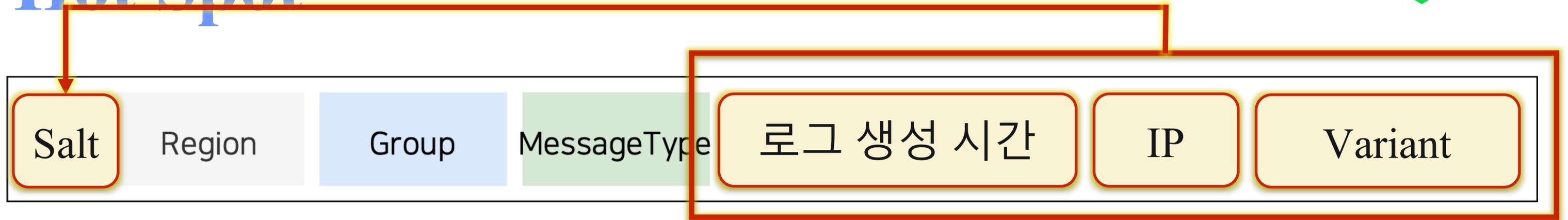
## HBase Table

R	korea	G	naver_search	M	query_log	Time	1571337770000...
R	korea	G	naver_search	M	query_log	Time	1571337770000...
R	korea	G	naver_search	M	query_log	Time	1571337790000...

Hot Spot

RowKey 앞부분이 동일

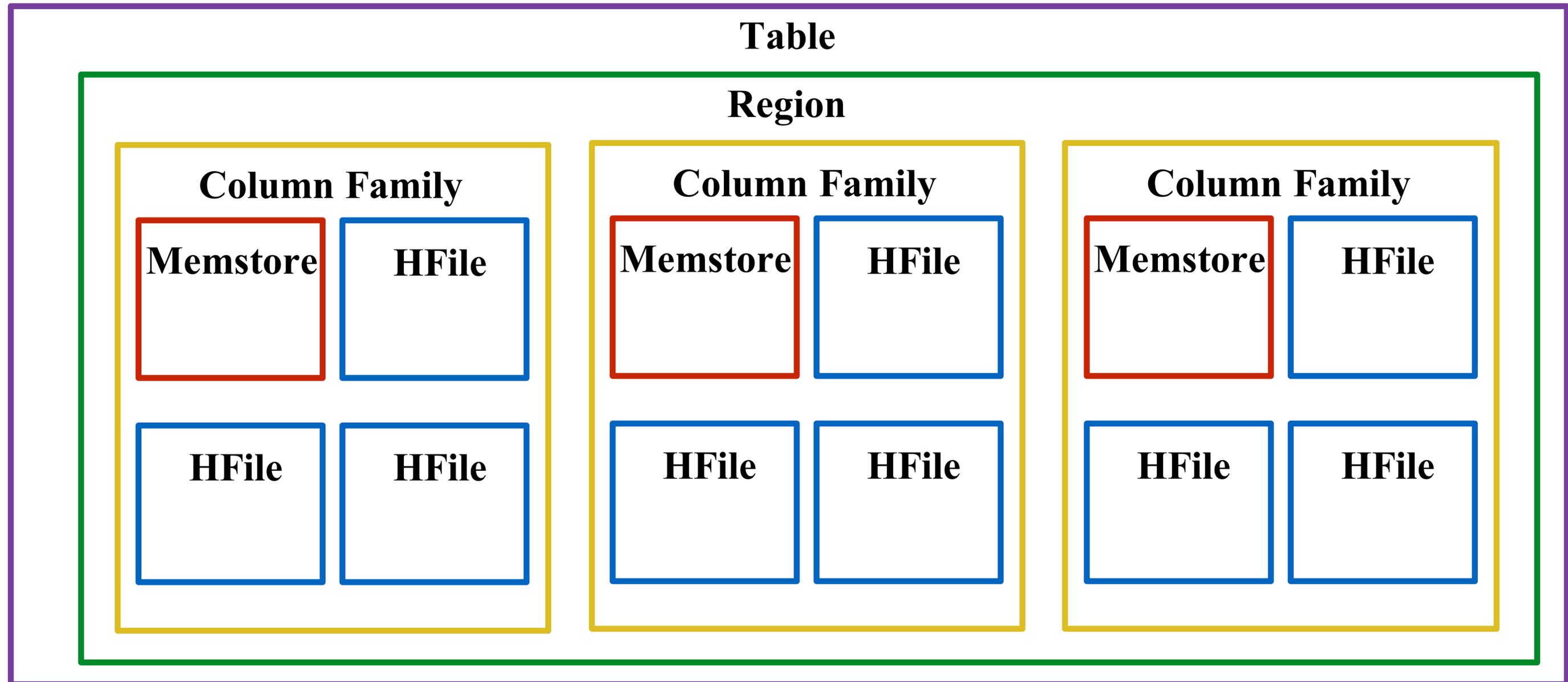
# Hot Spot



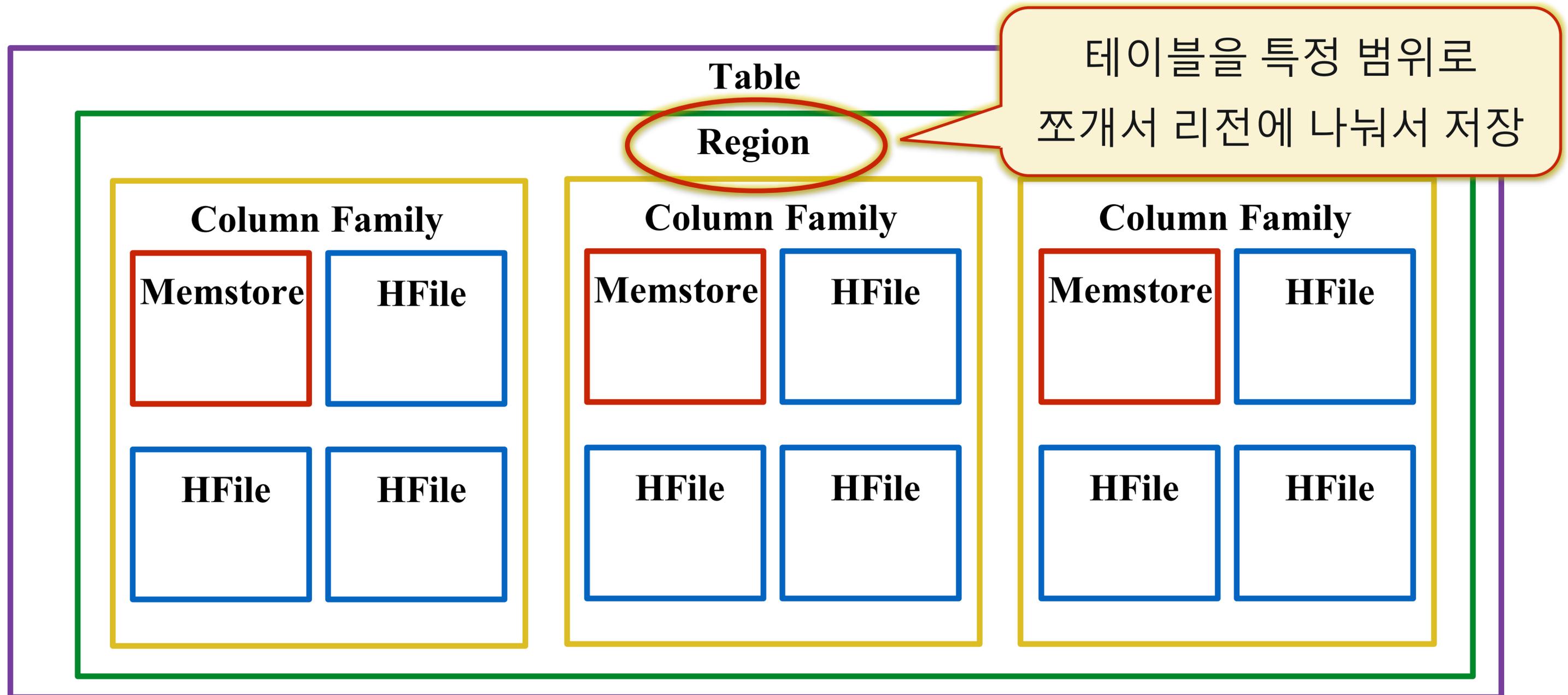
## HBase Table

R	korea	G	naver_search	M	query_log	Time	1571337770000...
R	korea	G	naver_search	M	query_log	Time	1571337780000...
R	korea	G	naver_search	M	query_log	Time	1571337790000...

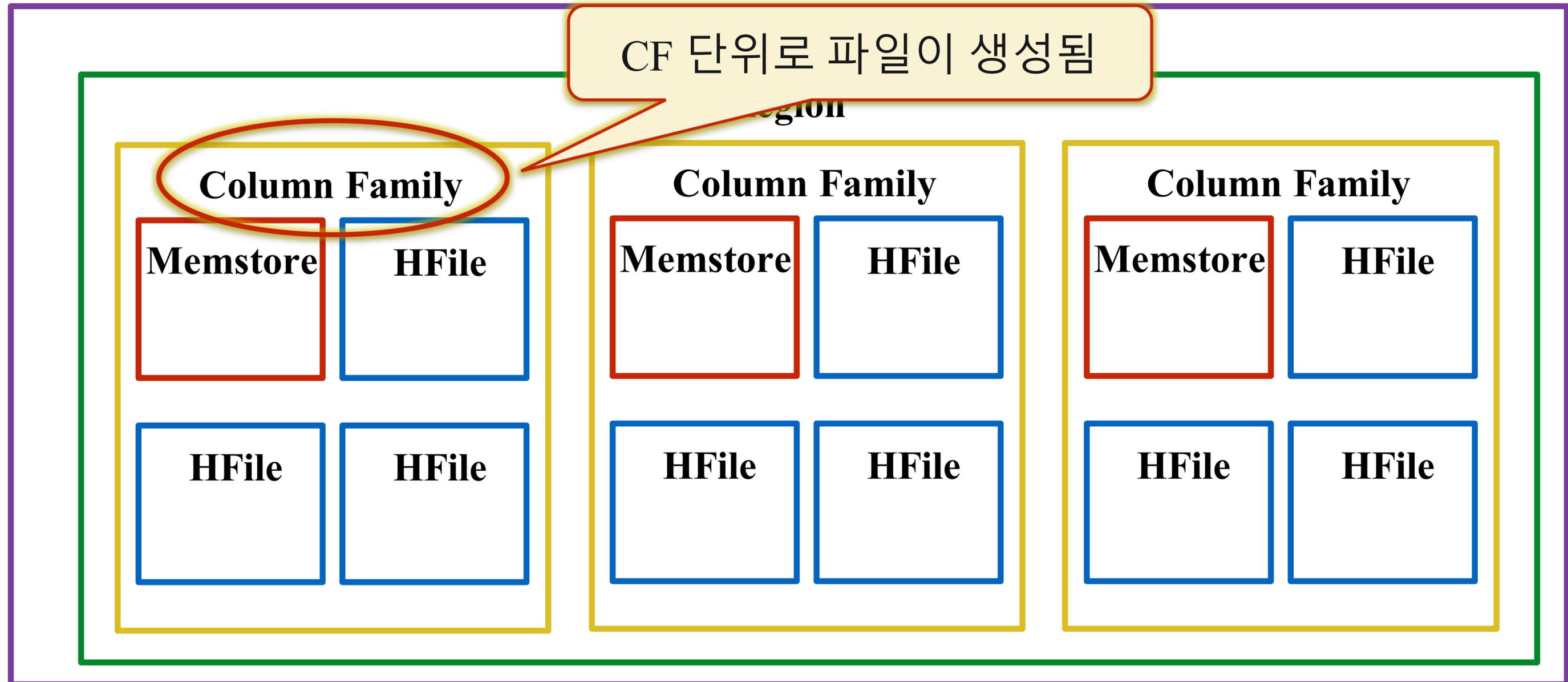
# 저장 공간을 용도별로 나눔



# 저장 공간을 용도별로 나눔



# 저장 공간을 용도별로 나눔



# 저장 공간을 용도별로 나눔

데이터를 읽는 패턴과 포맷이 비슷하면  
같은 CF로 묶는 것이 효율적

**Column Family**

**Memstore**

**HFile**

**HFile**

**HFile**

**Column Family**

**Memstore**

**HFile**

**HFile**

**HFile**

**Column Family**

**Memstore**

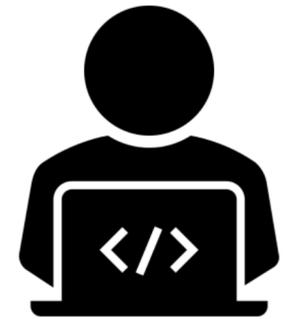
**HFile**

**HFile**

**HFile**

# 저장 공간을 용도별로 나눔

1571138636135	"GET /search.naver?query=데뷰	HTTP/1.1"	200	Chrome
1571138651527	"GET /search.naver?query=데뷰2019	HTTP/1.1"	200	Whale
1571138659998	"GET /search.naver?query=DataSuite	HTTP/1.1"	200	Whale
1571138665826	"GET /search.naver?query=DataStore	HTTP/1.1"	200	Whale



소비자

원본 로그가 필요해

# 저장 공간을 용도별로 나눔

1571138636135	"GET /search.naver?query=데뷰	HTTP/1.1"	200	Chrome
1571138651527	"GET /search.naver?query=데뷰2019	HTTP/1.1"	200	Whale
1571138659998	"GET /search.naver?query=DataSuite	HTTP/1.1"	200	Whale
1571138665826	"GET /search.naver?query=DataStore	HTTP/1.1"	200	Whale

보안정보

HTTP method

검색 쿼리

프로토콜

HTTP Status

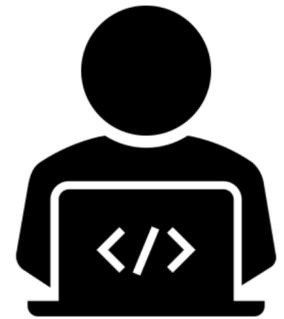
User Agent

# 저장 공간을 용도별로 나눔

1571138636135	"GET /search.naver?query=데뷰	HTTP/1.1"	200	Chrome
1571138651527	"GET /search.naver?query=데뷰2019	HTTP/1.1"	200	Whale
1571138659998	"GET /search.naver?query=DataSuite	HTTP/1.1"	200	Whale
1571138665826	"GET /search.naver?query=DataStore	HTTP/1.1"	200	Whale

검색 쿼리

검색 쿼리만 필요해



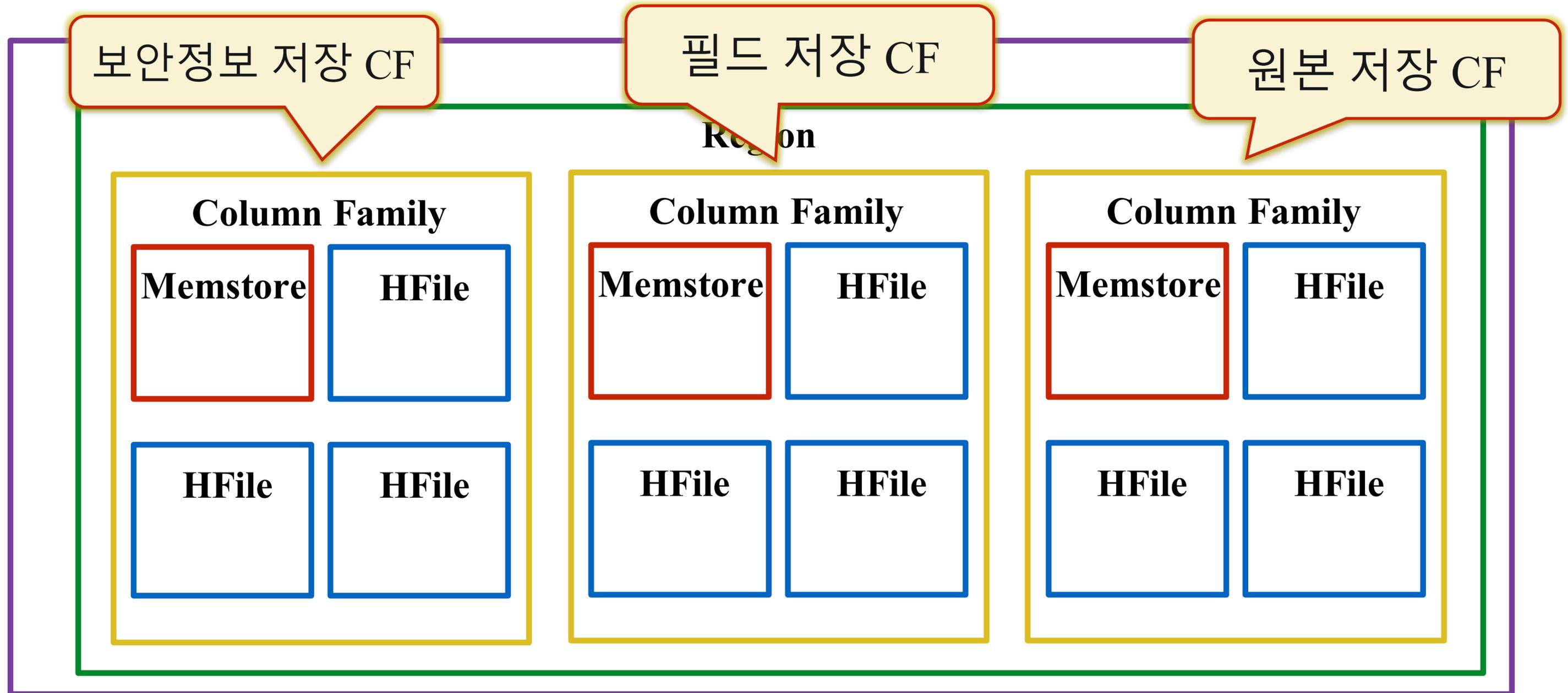
소비자

# 저장 공간을 용도별로 나눔

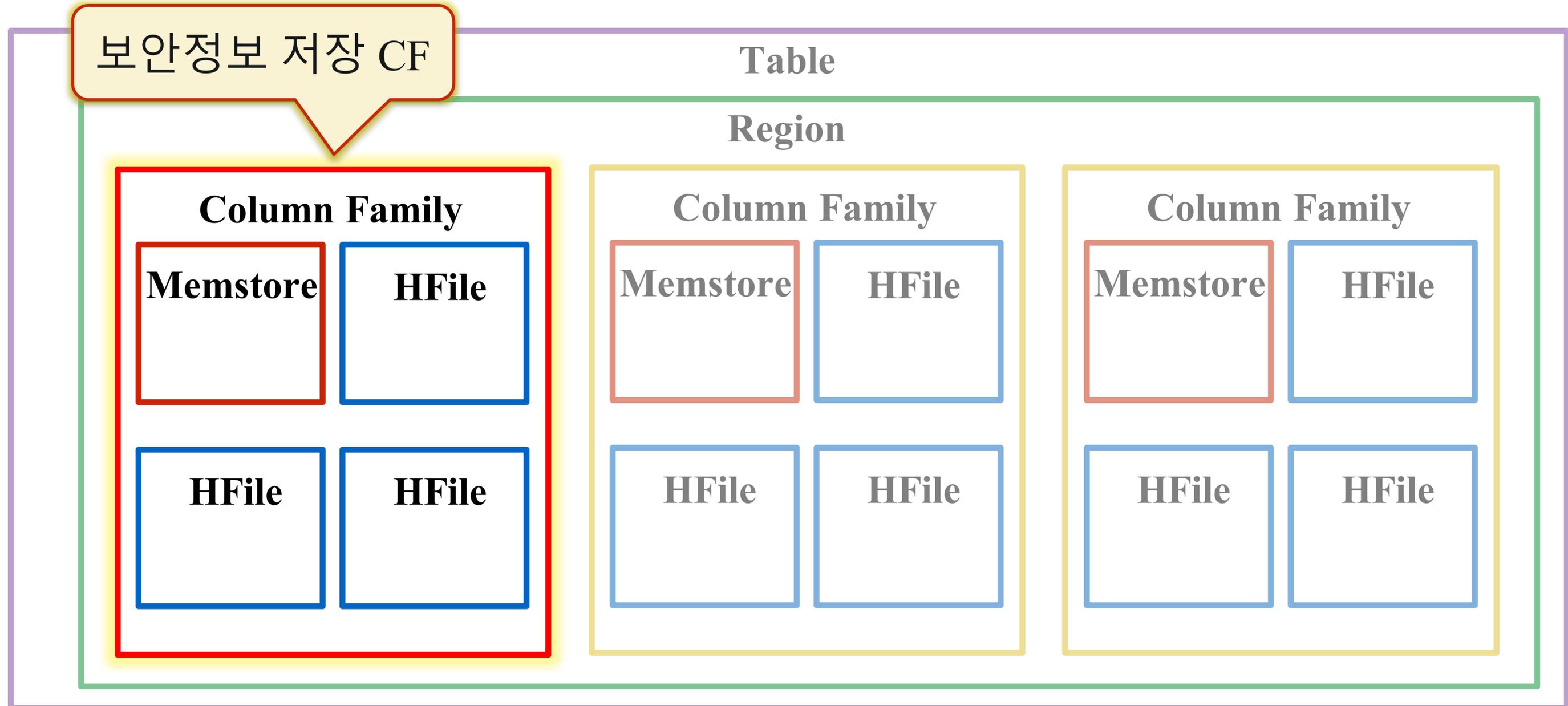
1571138636135	"GET /search.naver?query=데뷰	HTTP/1.1"	200	Chrome
1571138651527	"GET /search.naver?query=데뷰2019	HTTP/1.1"	200	Whale
1571138659998	"GET /search.naver?query=DataSuite	HTTP/1.1"	200	Whale
1571138665826	"GET /search.naver?query=DataStore	HTTP/1.1"	200	Whale

보안정보

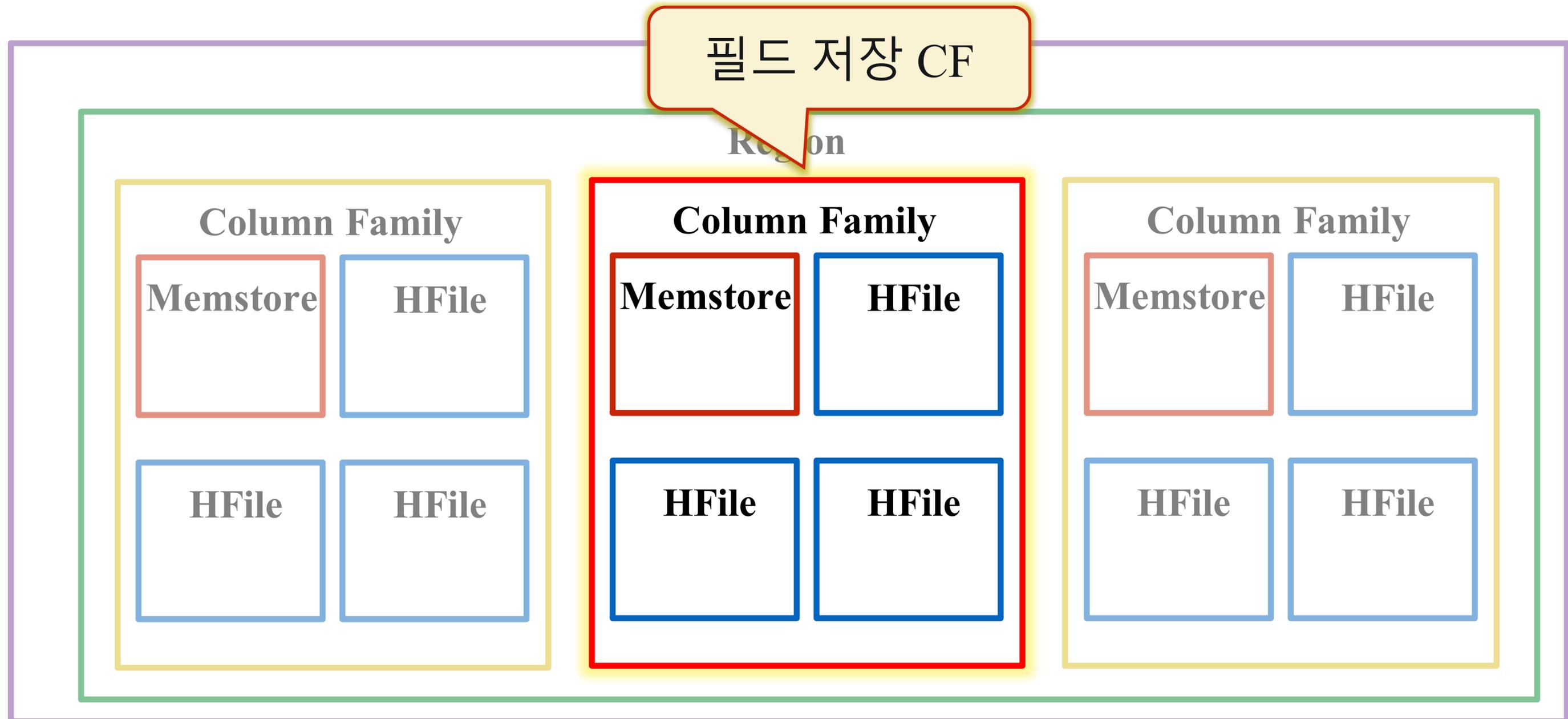
# 저장 공간을 용도별로 나눔



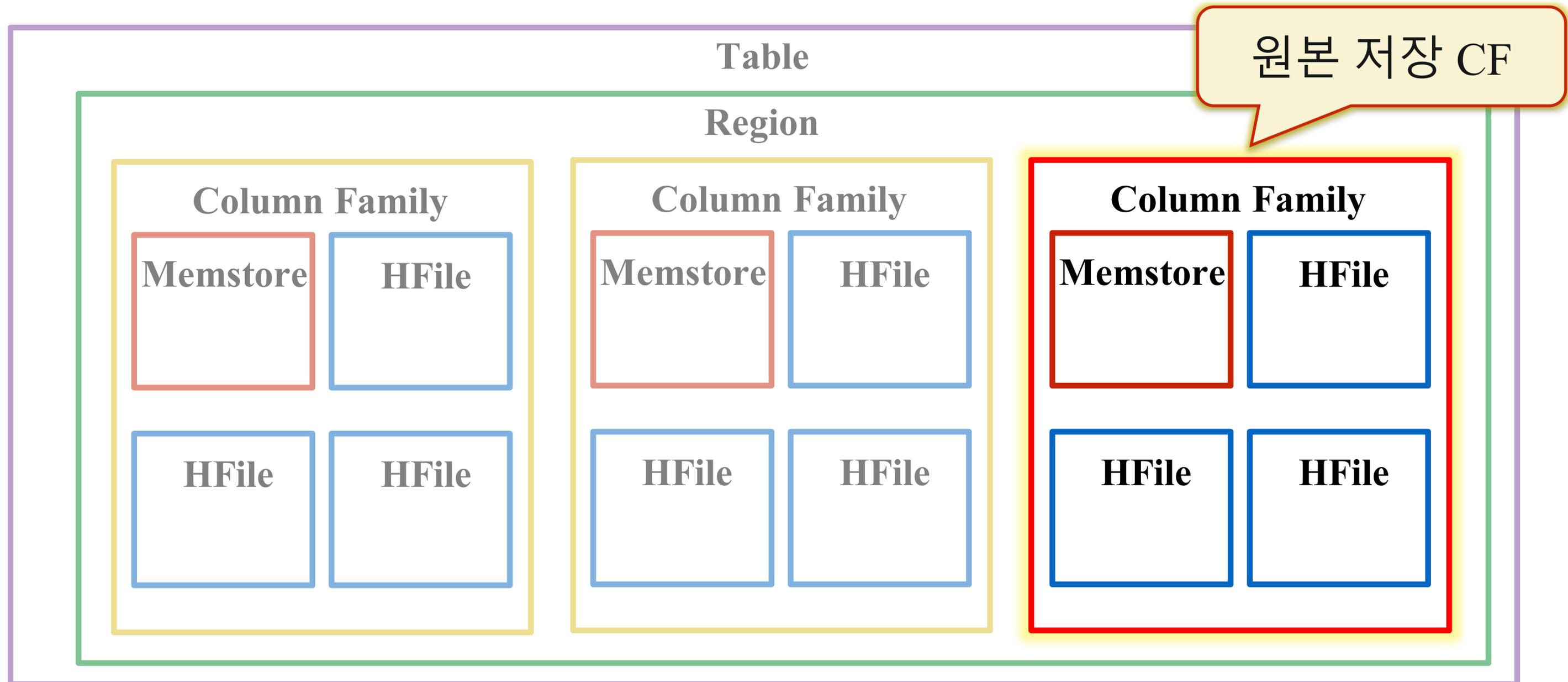
# 저장 공간을 용도별로 나눔



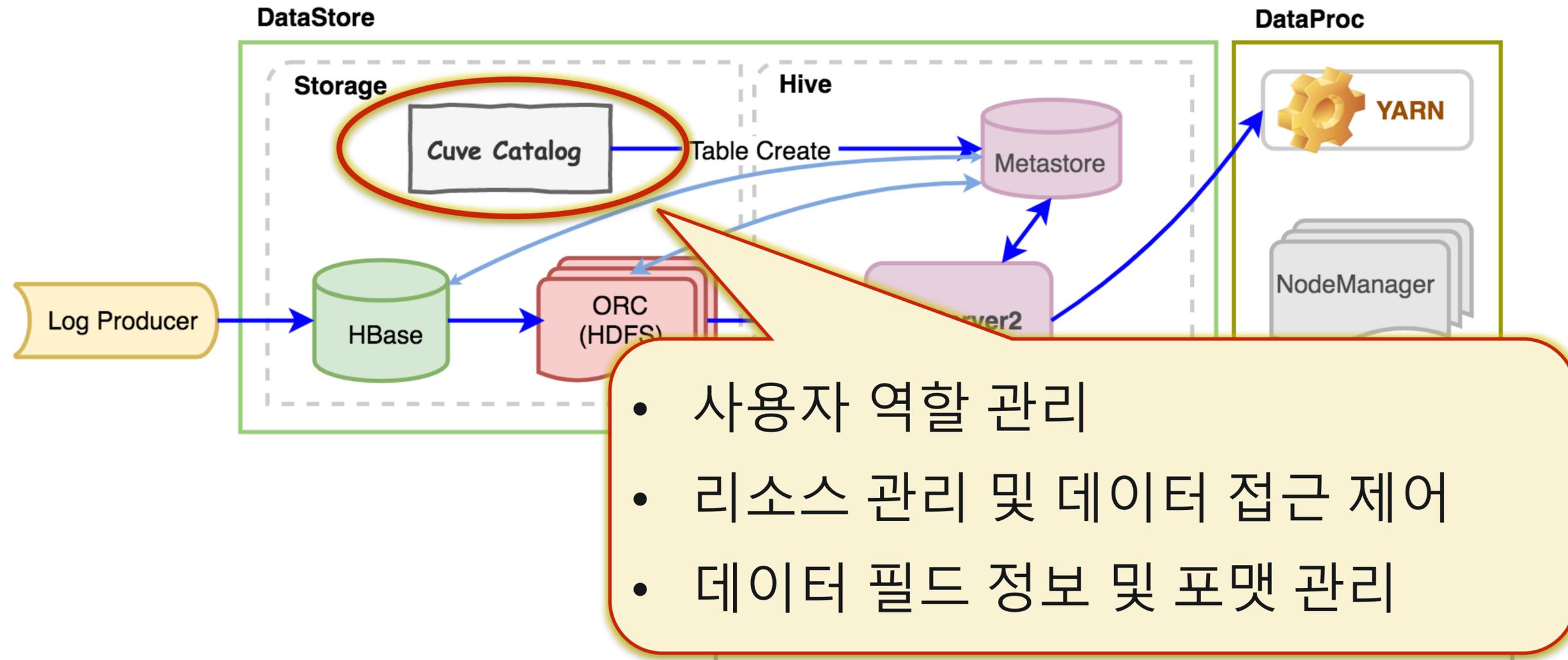
# 저장 공간을 용도별로 나눔



# 저장 공간을 용도별로 나눔



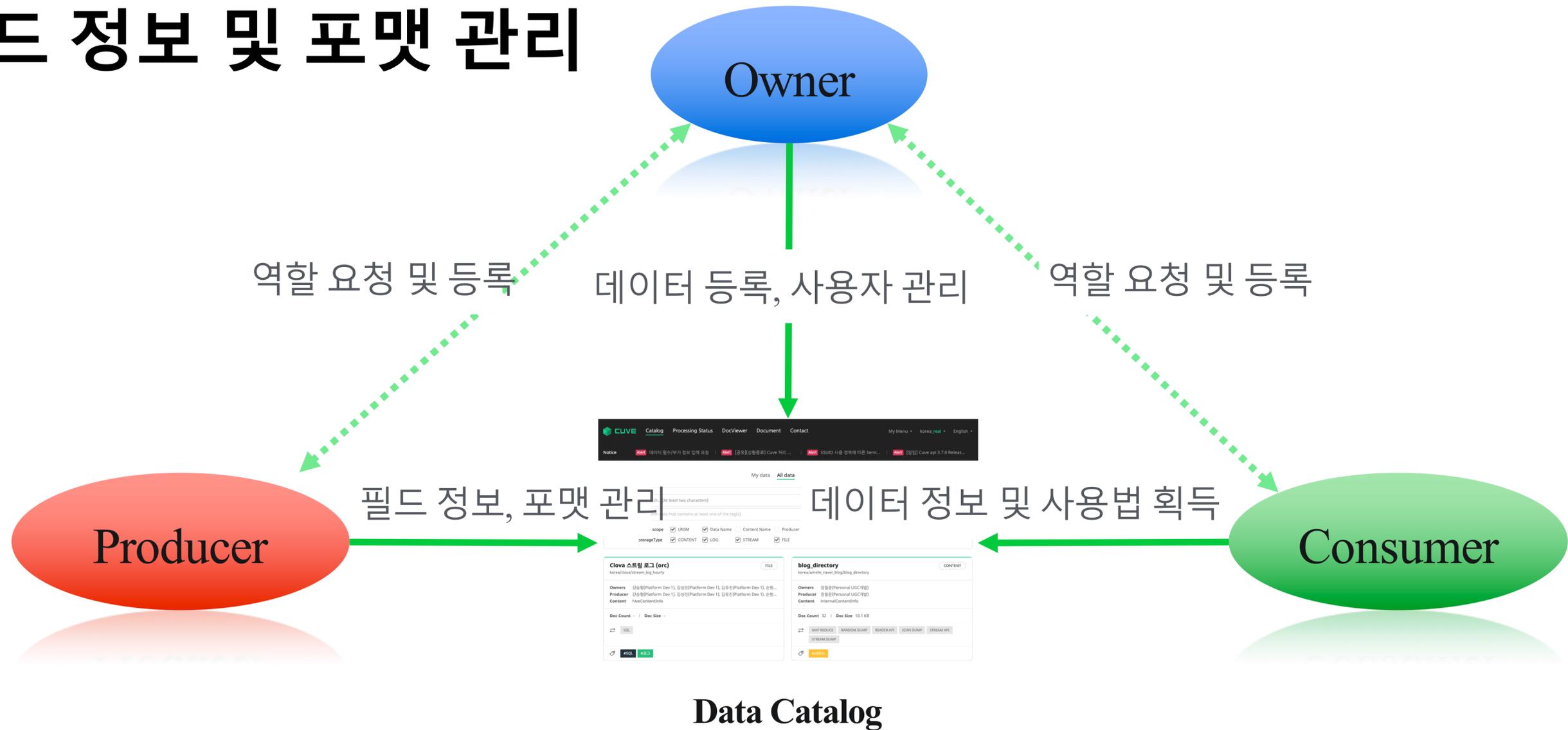
# 로그 파이프라인



# Data Catalog

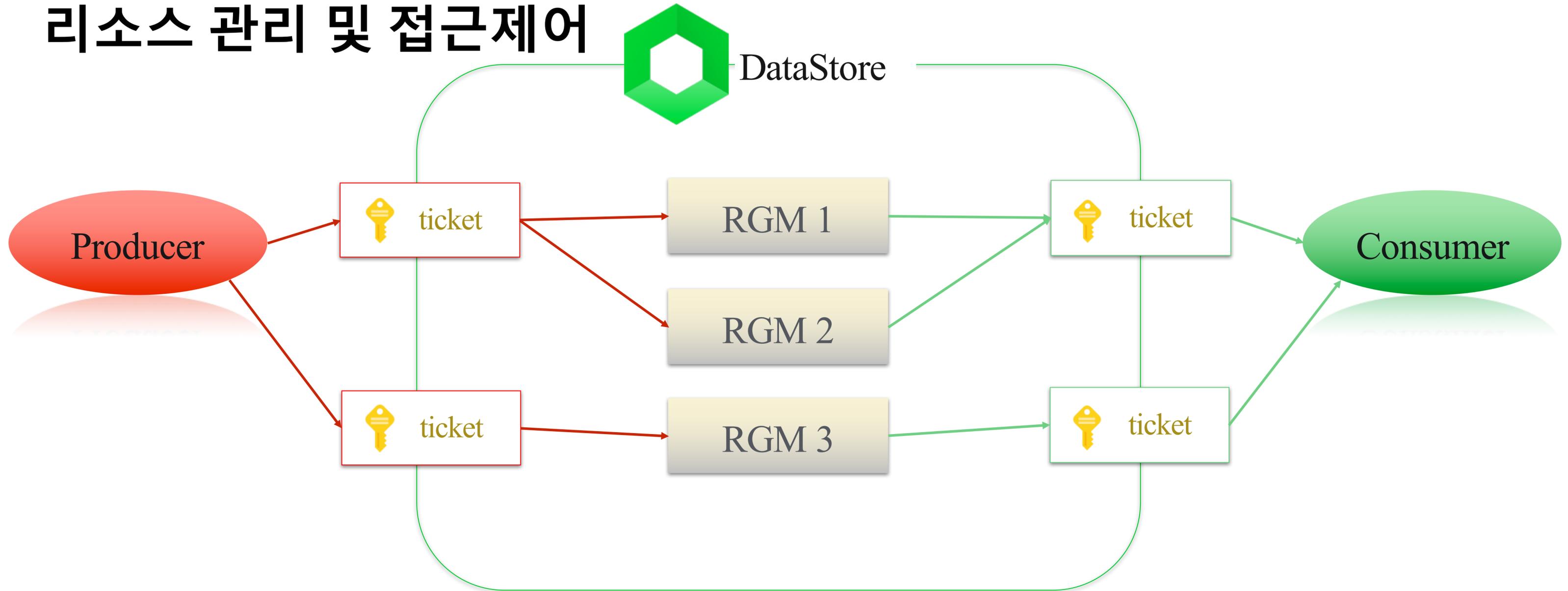
## 사용자 역할 관리

## 데이터 필드 정보 및 포맷 관리



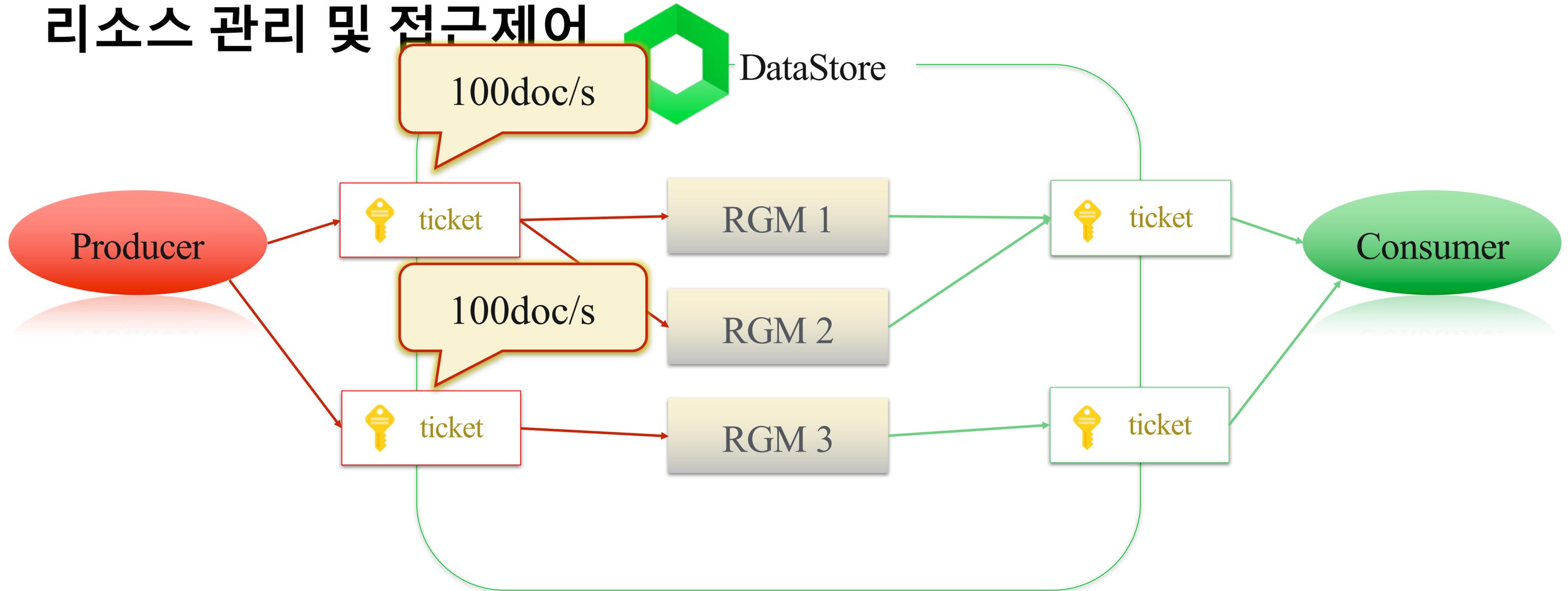
# 티켓

## 리소스 관리 및 접근제어



# 티켓

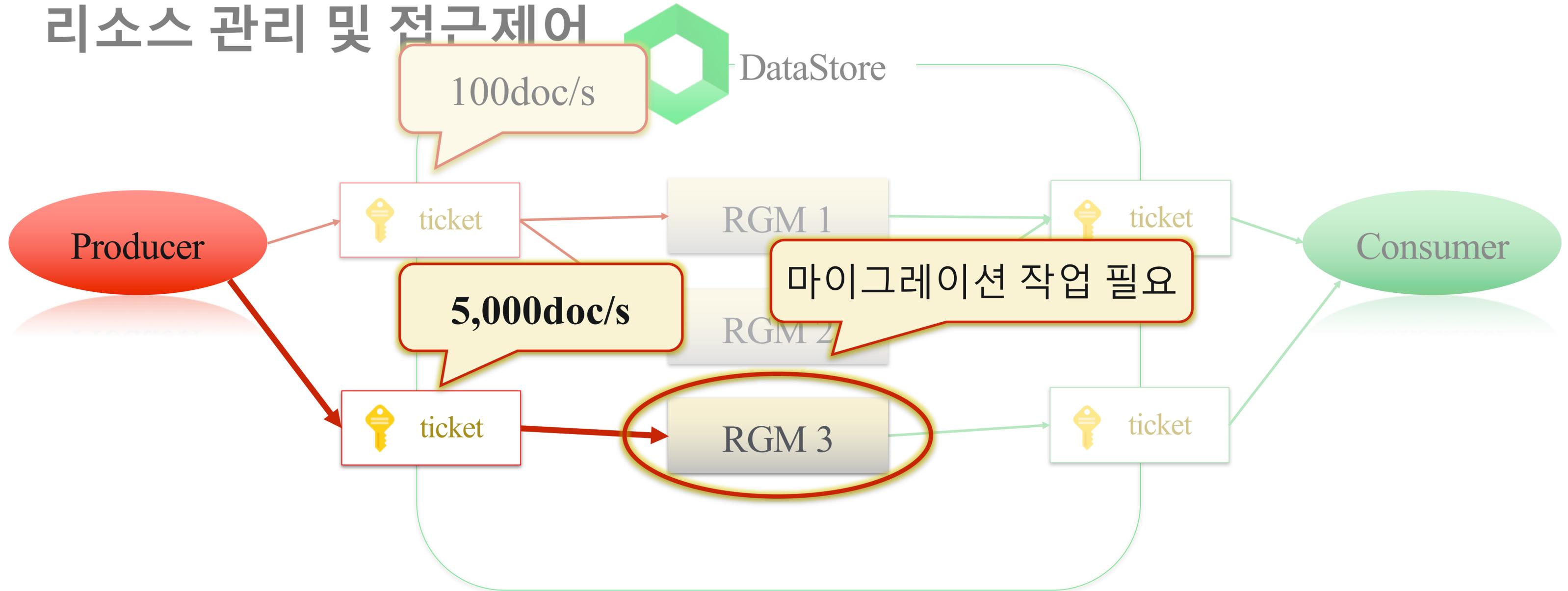
## 리소스 관리 및 접근제어





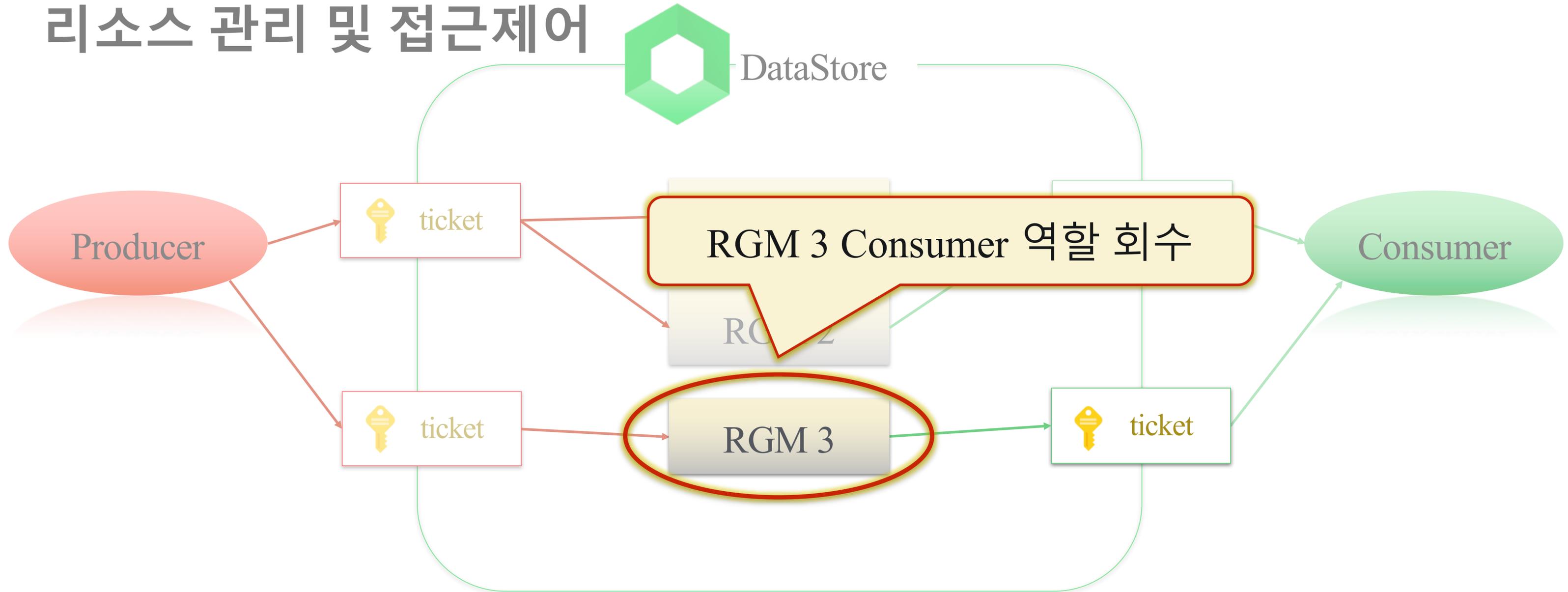
# 티켓

## 리소스 관리 및 접근제어



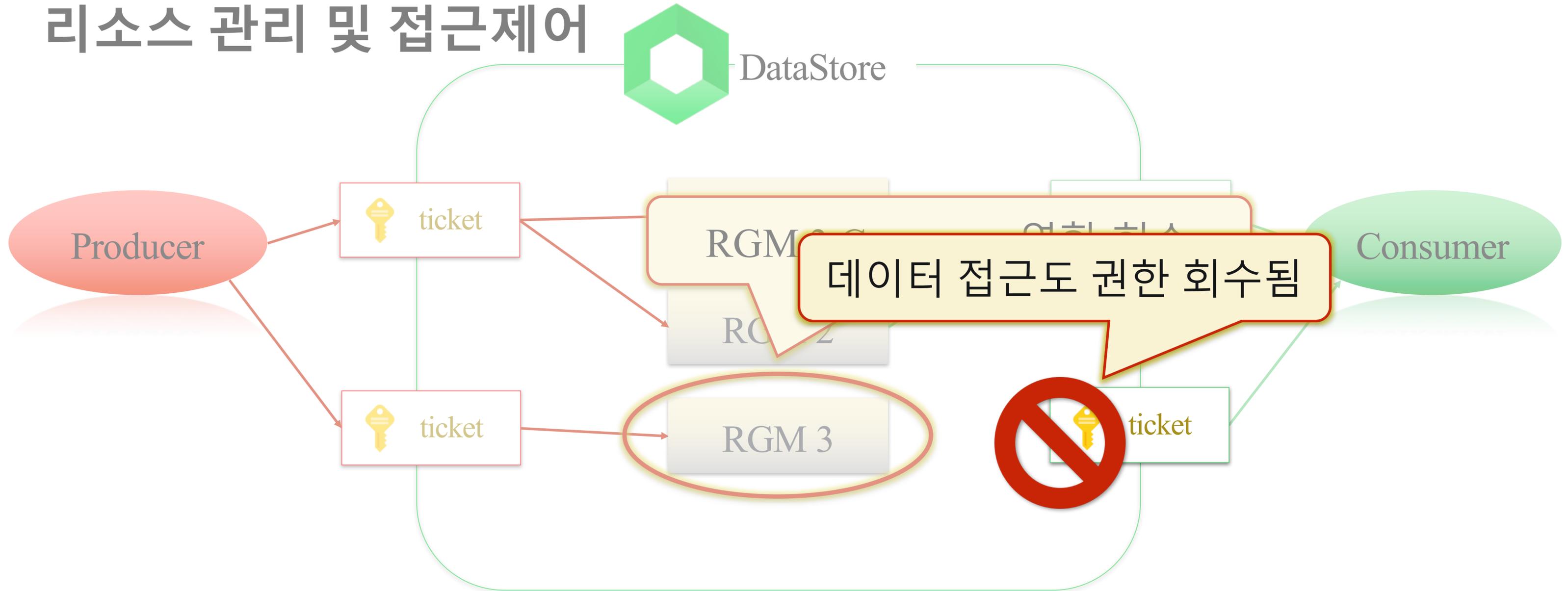
# 티켓

## 리소스 관리 및 접근제어



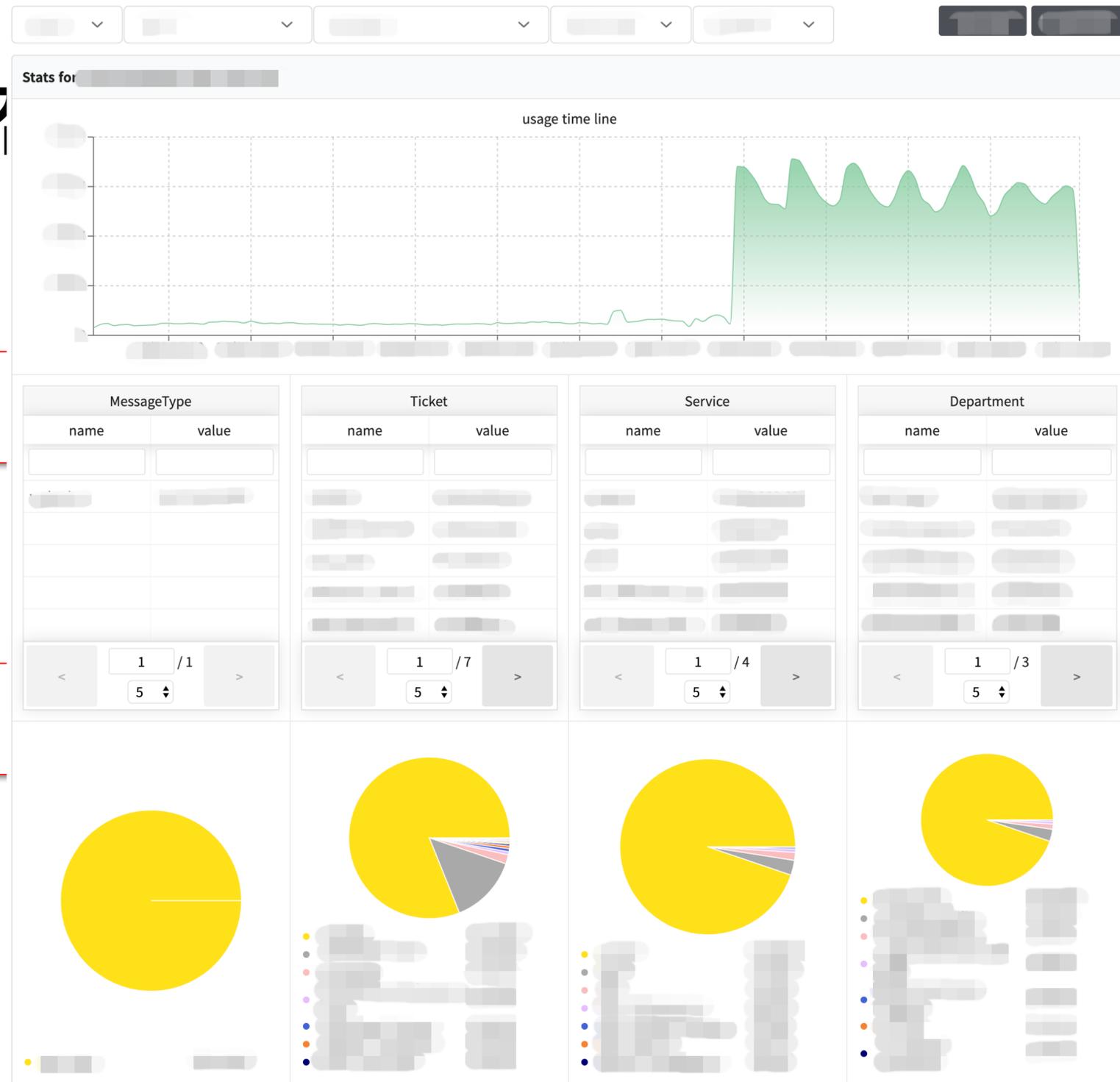
# 티켓

## 리소스 관리 및 접근제어

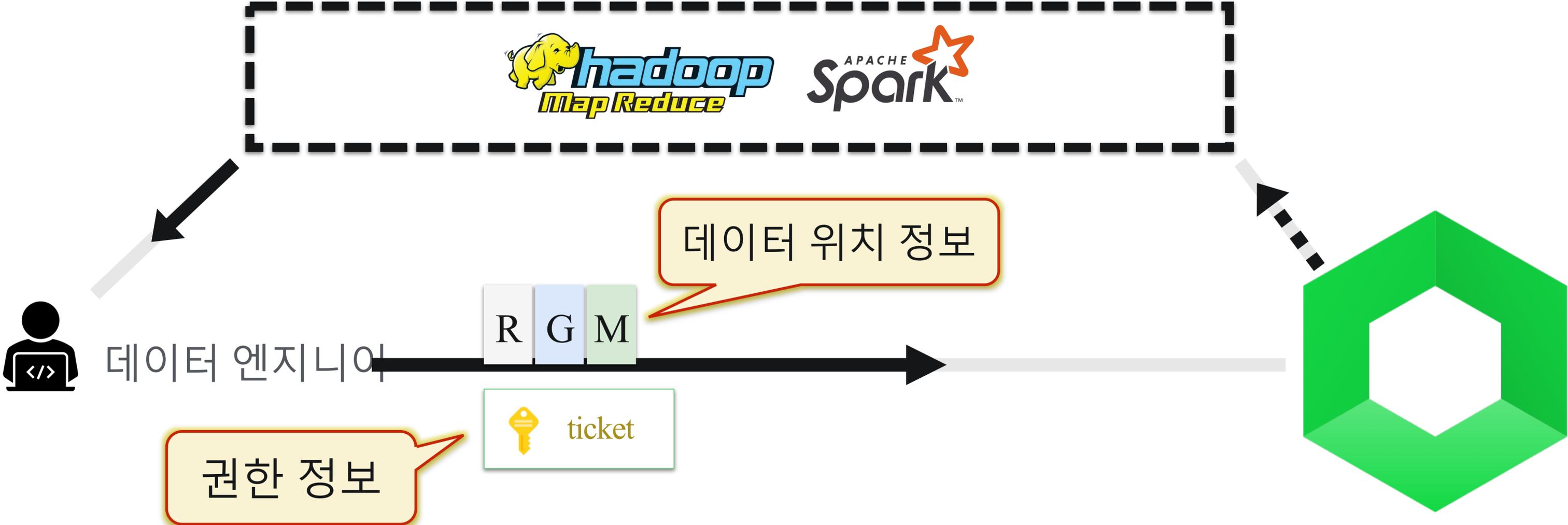


# 티켓

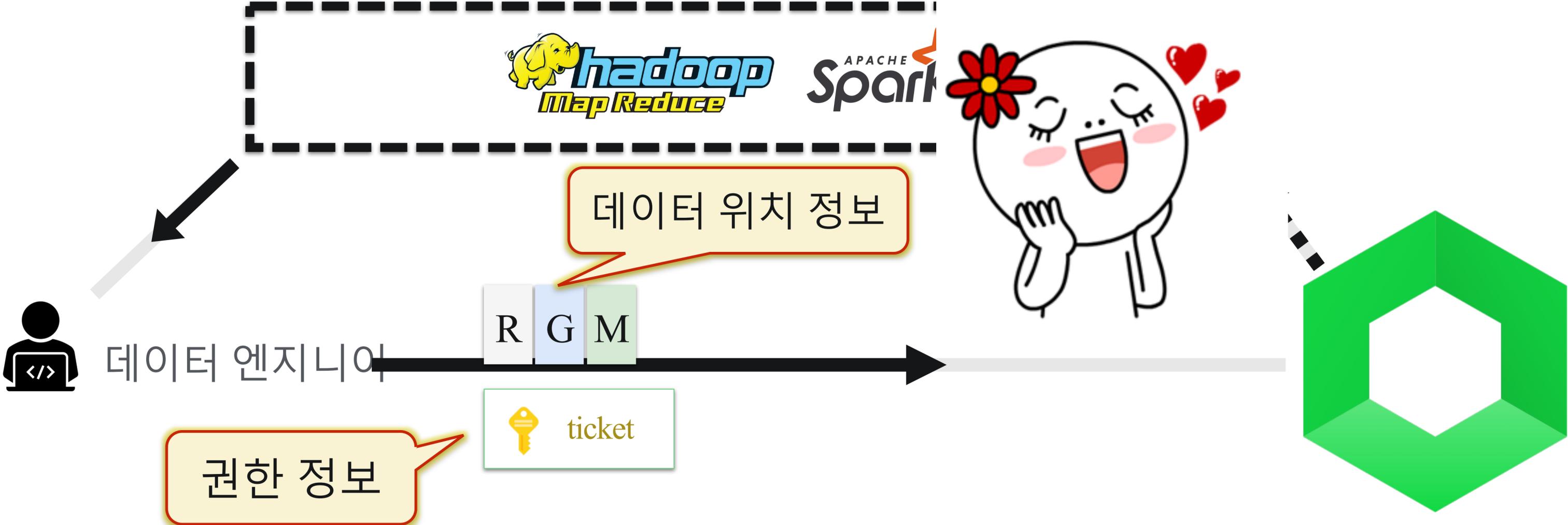
## 리소스 관리 및



# DataStore

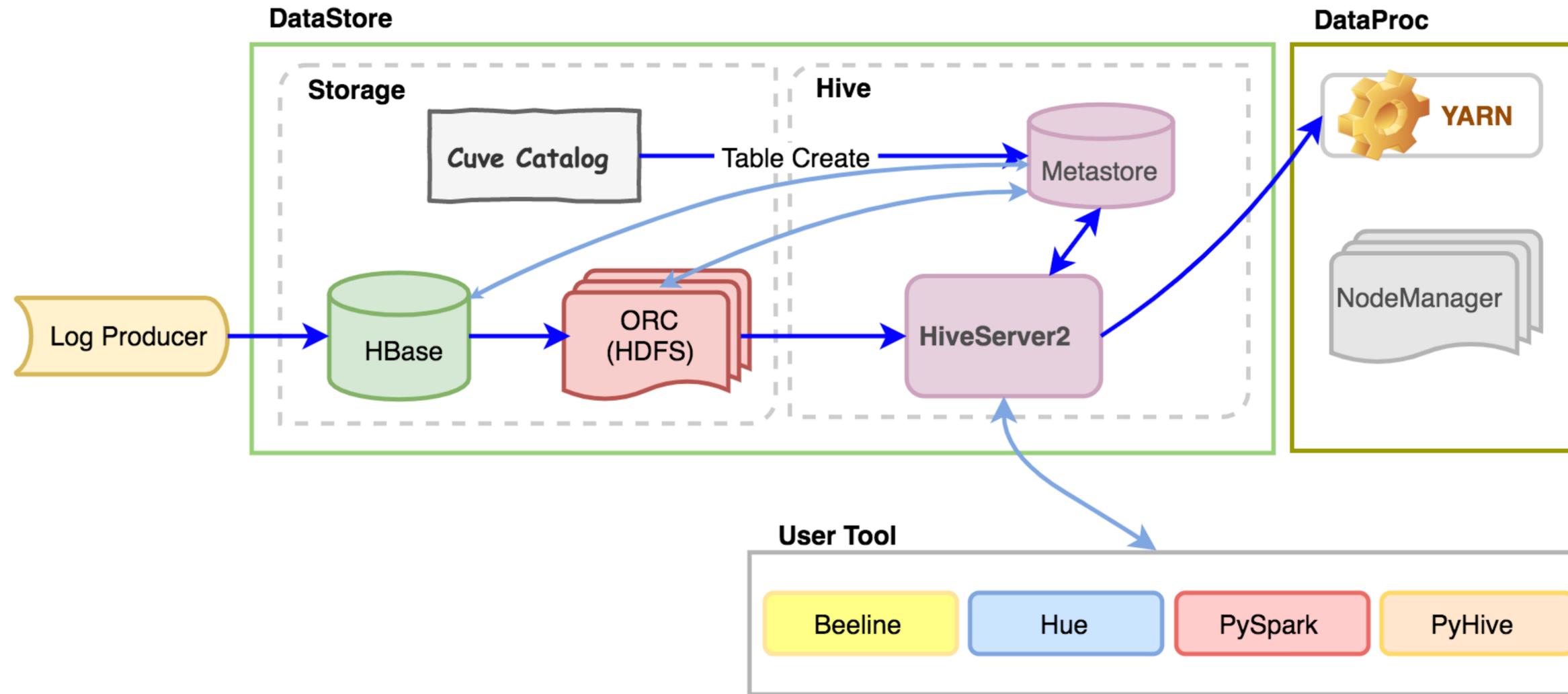


# DataStore



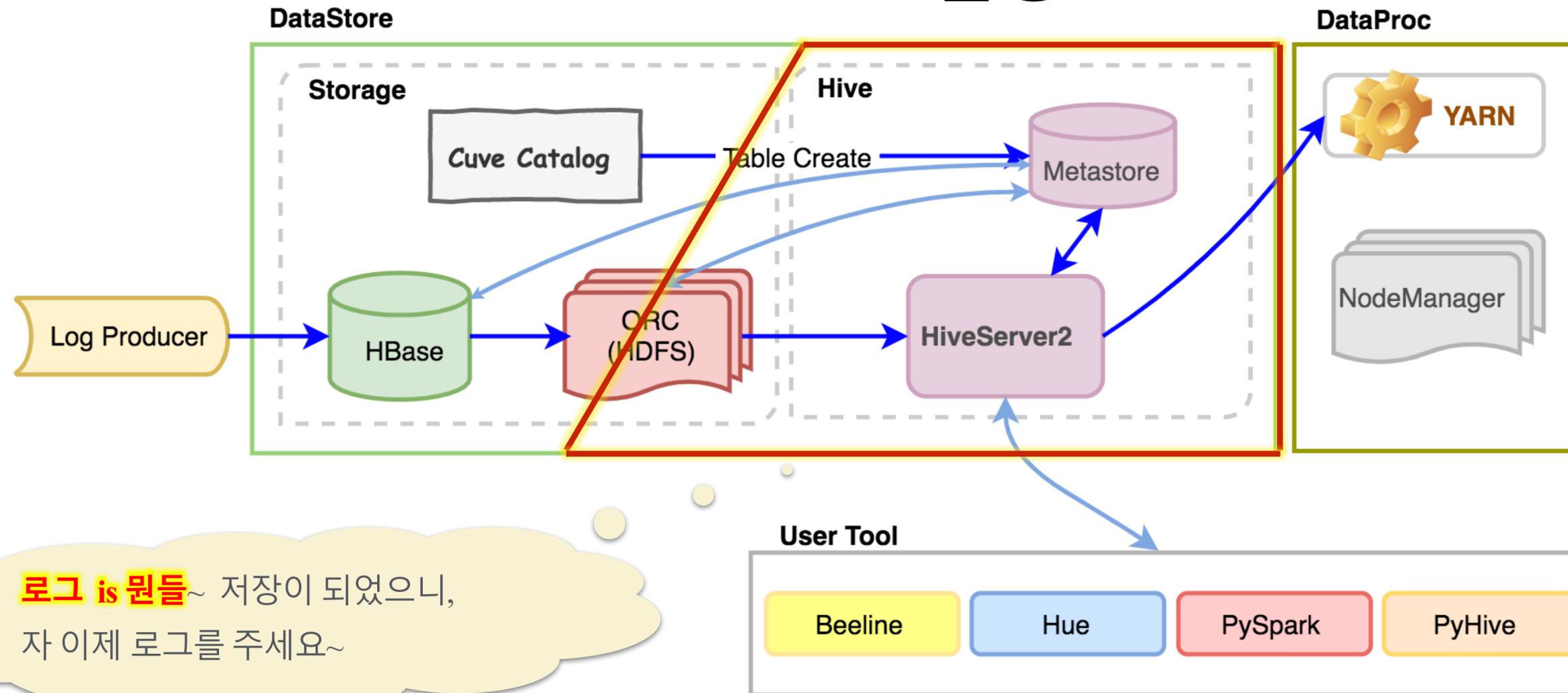
## 2. 로그 활용 이야기

# 로그 파이프라인



# 로그 파이프라인

## 활용



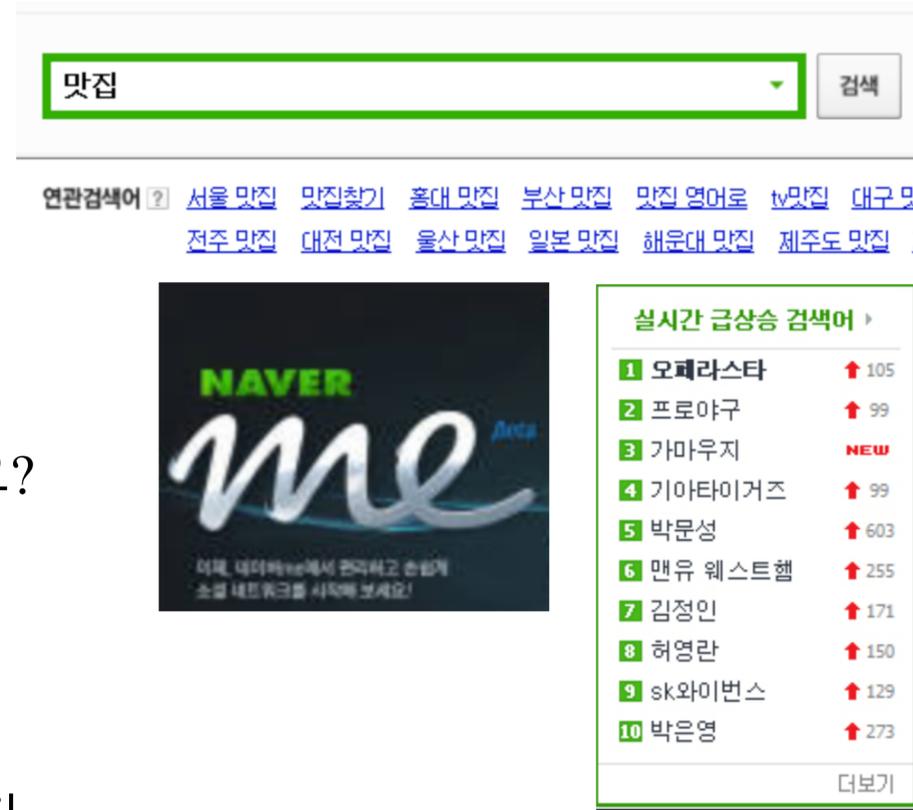
로그 is 원들~ 저장이 되었으니,  
자 이제 로그를 주세요~

# 네이버 검색 로그

네이버에서 검색 로그란?  
새로운 서비스의 시작  
사용자 반응의 바로미터

이러한 로그는 과연 얼마나 DataStore에 쌓이고 있을까요?  
모바일 통합검색 로그량만 하루 **수십억** 건  
로그 사이즈는 **수TB** 정도

방대한 양의 로그를 매일 같이 필요한 사람들이 있습니다.



1	오페라스타	↑ 105
2	프로야구	↑ 99
3	가마우지	NEW
4	기아타이거즈	↑ 99
5	박문성	↑ 603
6	맨유 웨스트햄	↑ 255
7	김정인	↑ 171
8	허영란	↑ 150
9	sk와이번스	↑ 129
10	박은영	↑ 273



AiRS 추천 뉴스 <sup>Beta</sup> 를 소개합니다 > ✕

웨스트브룩, 시즌 40호 트리플-더블...6경기 연속

32개월 기다린 유소연, '4벌타' 톰슨보다 간절했다

[U-20 명단발표] 월드컵 전 마지막 소집...백승호-이승...

# 로그를 찾는 사람들

서비스 개발자

검색 모델링

높으신 분들

Data Scientist

기획/전략

보안

...

# 로그 활용



시도 때도 없이 쏟아 지는 로그 분석 의뢰들이 있습니다.

지난 달 십대가 검색한 '**~뜻**' 검색어 중 월간 QC 30 이상인 검색어만 추출

2018년 한 해 **이미지 탭검** 상위 검색어/탭검 진입경로 분석, 썸네일 클릭하지 않고 다른 영역으로 넘어가는 case

통검 **개편 이후** 웹검색 **소비 지표 변환**을 보기 위한 로그 분석

지난 **상반기** 상위 키워드에 대한 **광고 영역 CTR(클릭률)**

검색 상위 키워드에 대한 통합검색 **평균 컬렉션 랭킹 리스트**

1년간 **시간/연령/성별**에 따른 네이버 쇼핑 이용 패턴 분석

# 로그 활용

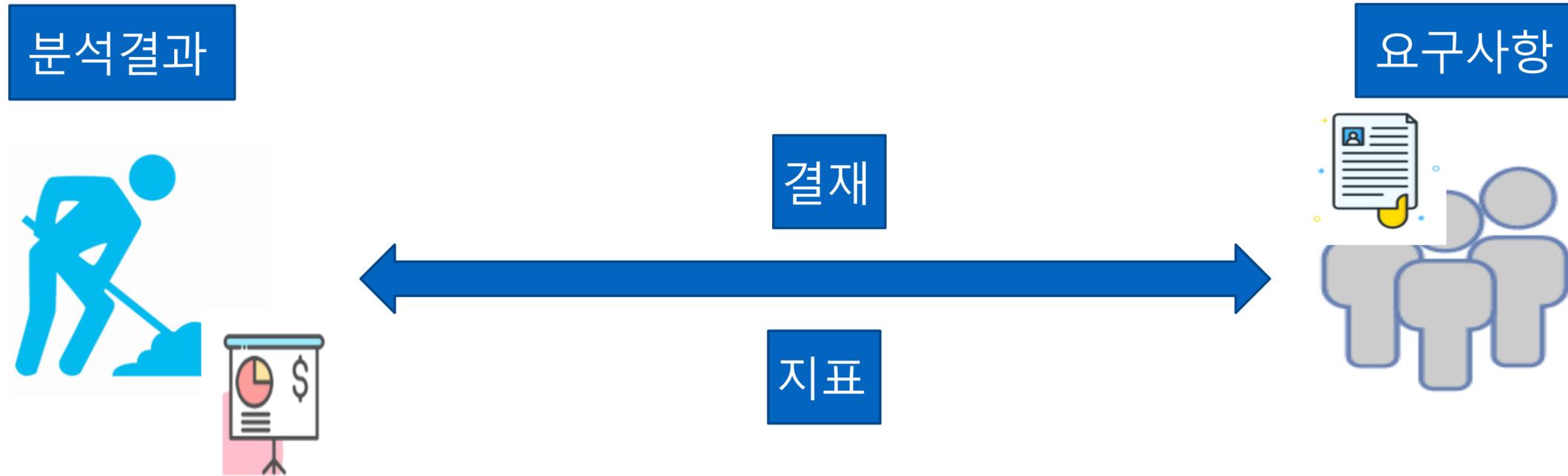
시도 때도 없이 쏟아 지는 분석 의뢰들이 있습니다.

지난 달 십대가 검색한 '**~뜻**' 검색어 중 월간 QC 30 이상인 검색어만 추출  
2018년 한 해 **이미지 탭검** 상위 검색어/탭검 진입경로 분석, 썸네일 클릭하지 않고 다른 영역으로 넘어가는 case  
통검 **개편 이후** 웹검색 **소비 지표 변환**을 보기 위한 로그 분석  
지난 **상반기** 상위 키워드에 대한 **광고 영역 CTR(클릭률)**  
검색 상위 키워드에 대한 통합검색 **평균 컬렉션 레킷**  
1년간 **시간/연령/성별**에 따른 네이

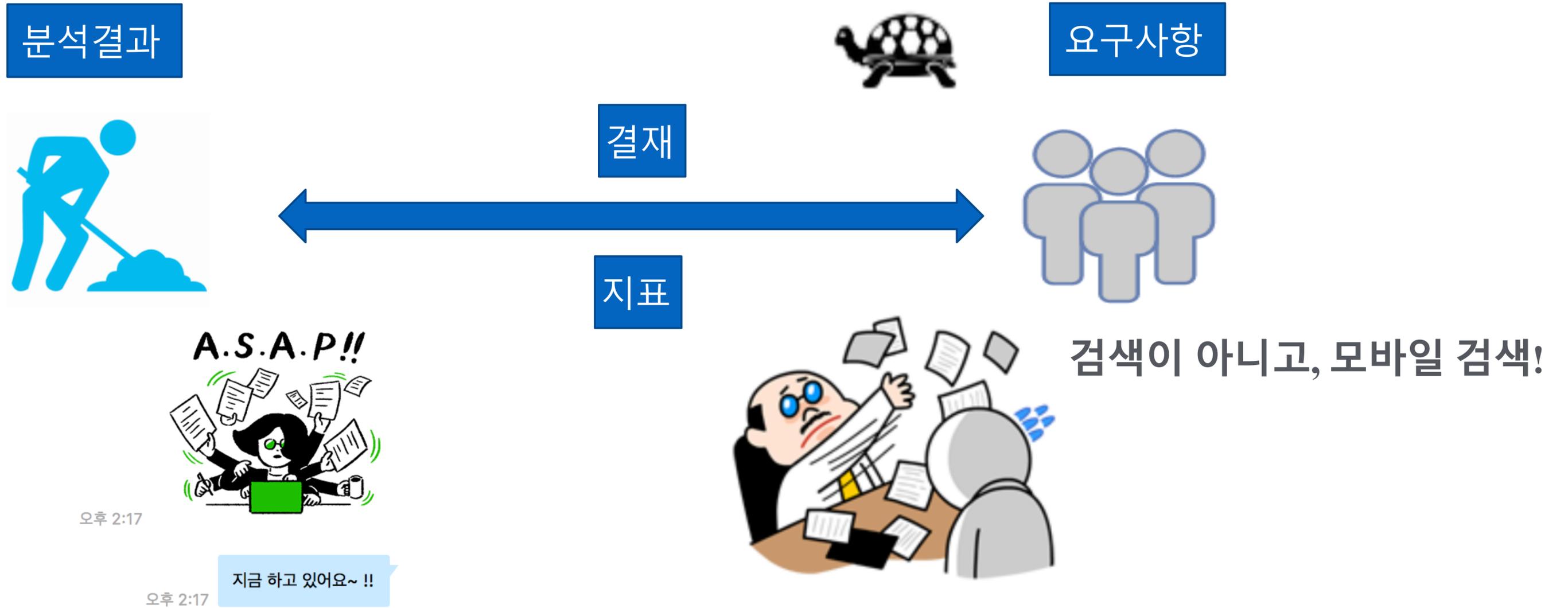
한 해 진행된 로그 분석 의뢰만 수백건



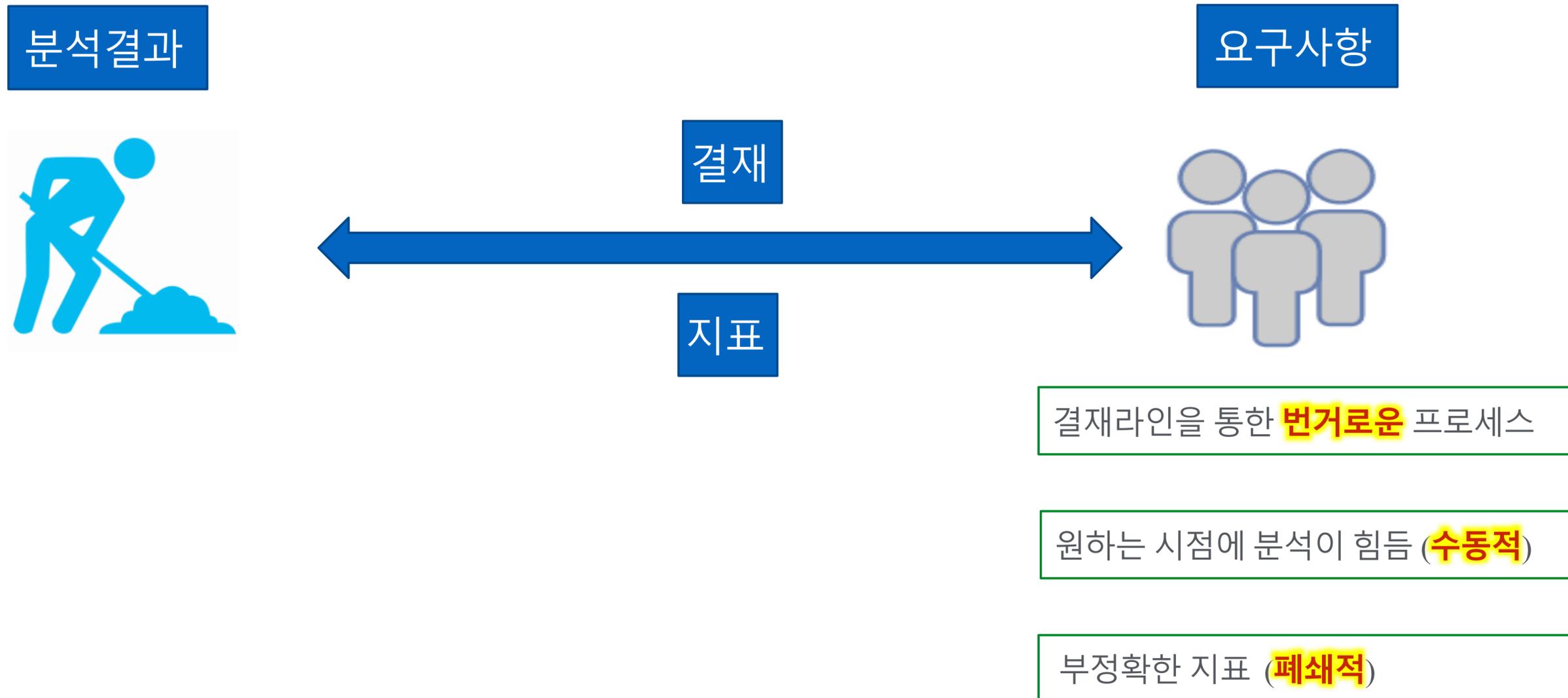
# 로그 활용의 문제점



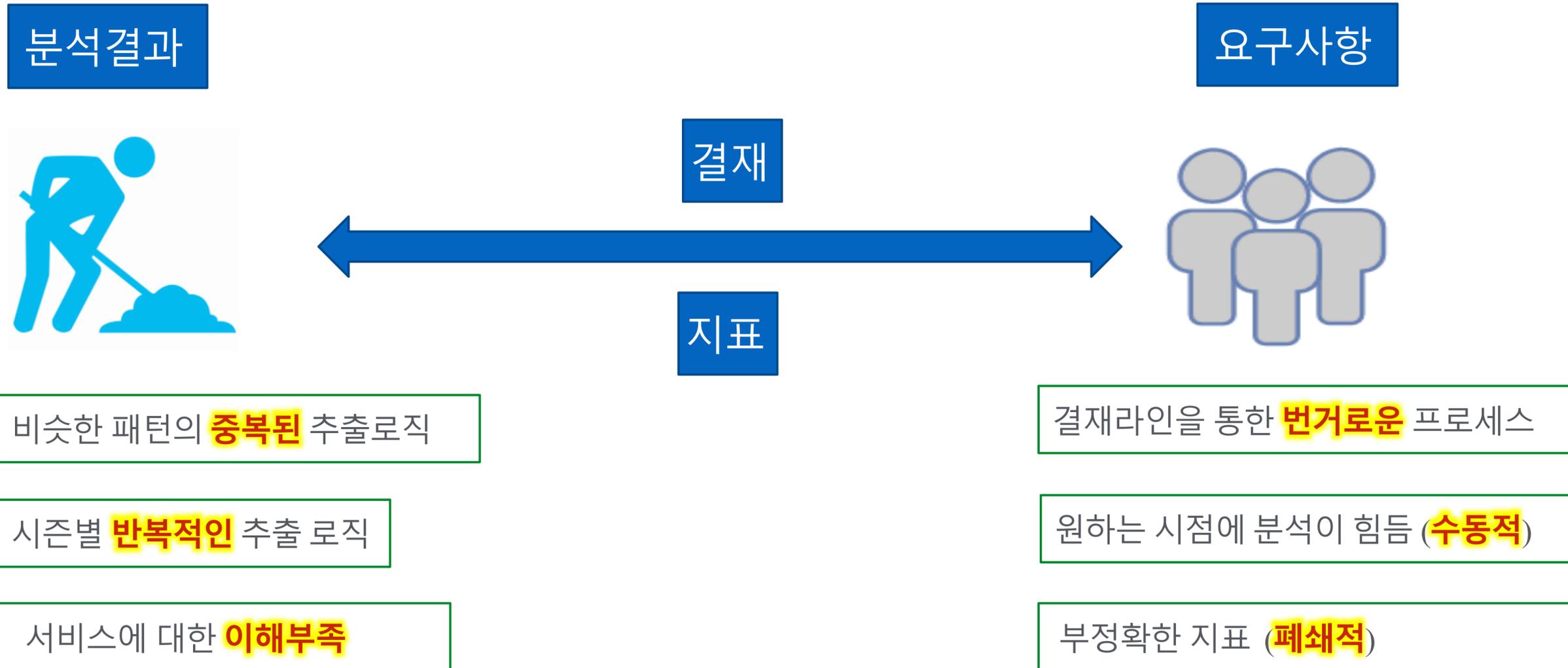
# 로그 활용의 문제점



# 로그 활용의 문제점



# 로그 활용의 문제점





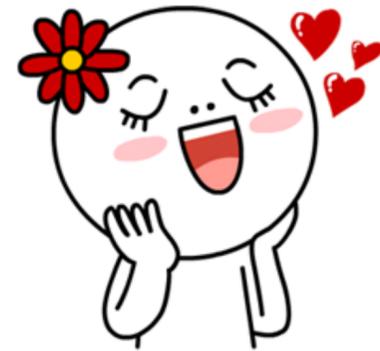




# 로그 사용성의 어려움

로그 생산자와 소비자도 관리되며  
접근이 허용된 사용자만 로그 사용이 가능하도록  
안전한 로그 관리 시스템이 준비되었다.

이제 사용만 하면 되는데,

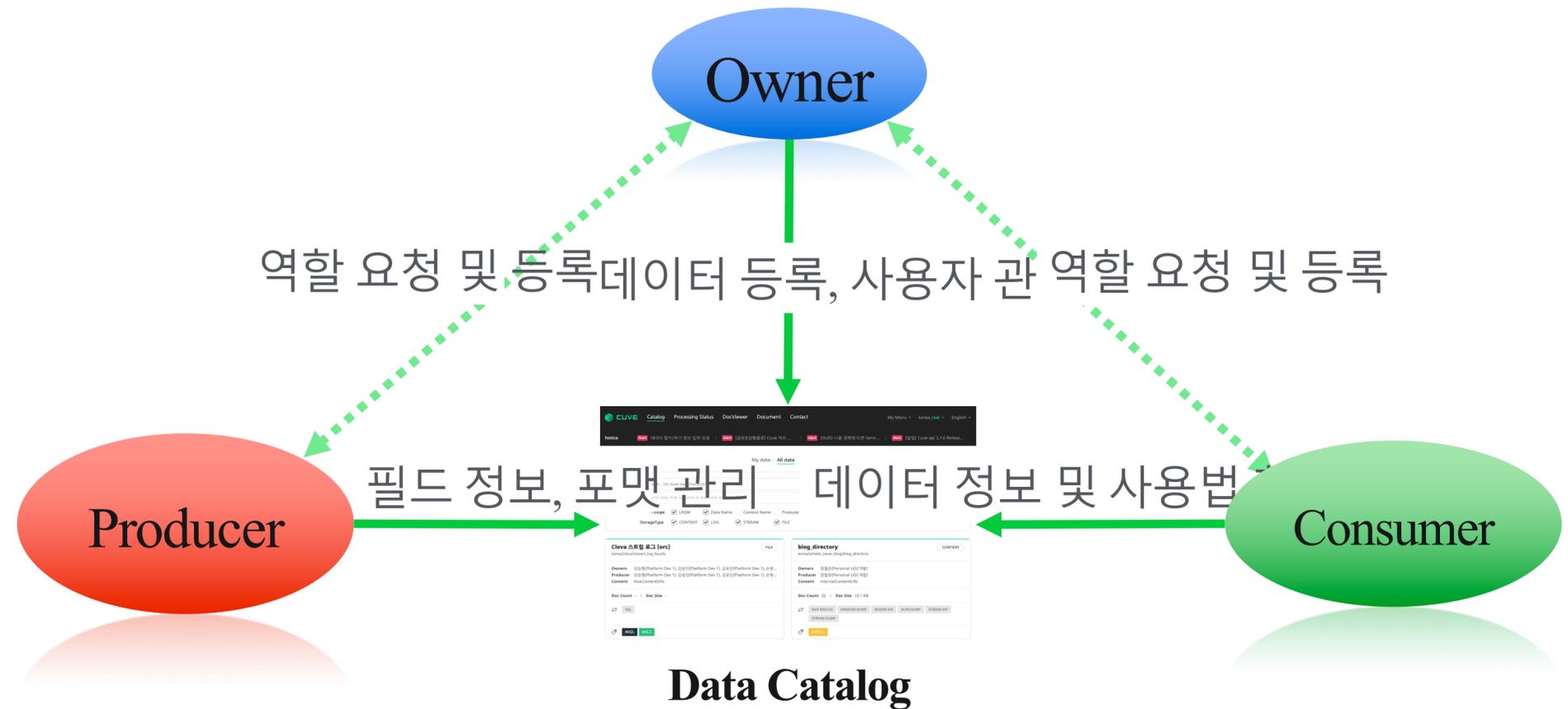


분산 파일 시스템(HDFS)

프로그래밍

터미널 환경

MapReduce

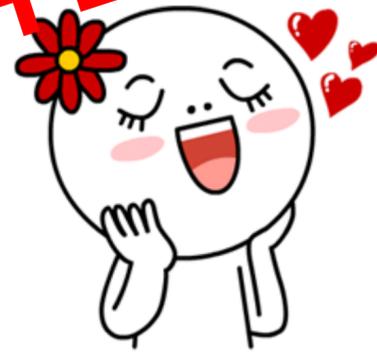


# 로그 사용성의 어려움

로그 생산자와 소비자도 관리되며  
 접근이 허용된 사용자만 로그 사용이 가능하도록  
 안전한 로그 관리 시스템이 준비되었다.

이제 사용만 하면 되는데,

**비개발 직군에게는 사용성의 한계**



분산 파일 시스템(HDFS)

프로그래밍

터미널 환경

MapReduce

Producer

Owner

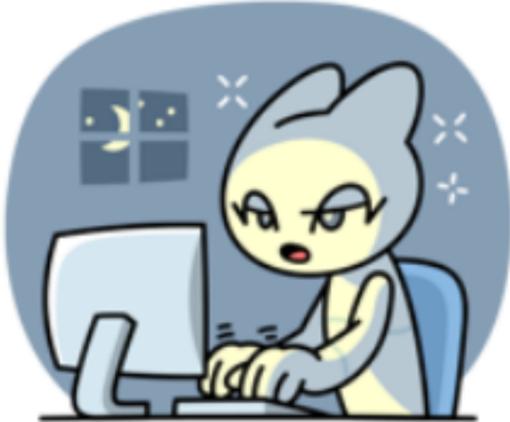
역할 요청 및 등록데이터 드로 사용자 과 역할 요청 및 등록

필드 정보, 포맷

**그림의 떡**



# SQL Interface

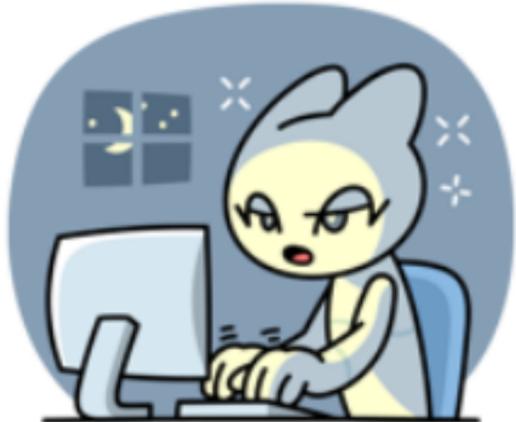


다양한 직군



다양한 사람들

# SQL Interface



다양한 직군



다양한 사람들

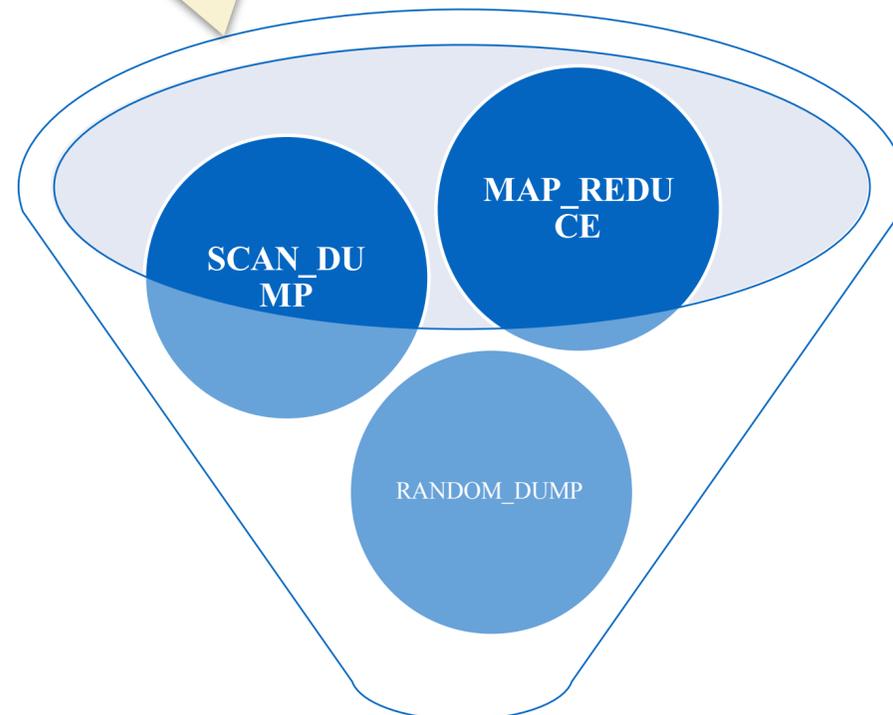


# SQL Interface

```
Mapper.py
stdin:
line.strip()
for item in line.split(delimiter):
    key = "
    try:
        key = re.search(r'%s(?:.*?)%s' % (prefix[0], prefix[1]),
item).group(1)
        if row.has_key(key):
            printOutputList(row, config, outputkeys)
            streamingCounter('Collection', 'items', 1)
            row.clear()
        row[key] = re.search(r'%s(?:.*?)%s' % (prefix[1], prefix[2]),
item).group(1)
    except AttributeError:
        continue
    ....
for line in sys.stdin:
    line = line.strip()
    for item in line.split(delimiter):
        key = "
        try:
            key = re.search(r'%s(?:.*?)%s' % (prefix[0], prefix[1]),
item).group(1)
            if row.has_key(key):
                printOutputList(row, config, outputkeys)
                streamingCounter('Collection', 'items', 1)
                row.clear()
            row[key] = re.search(r'%s(?:.*?)%s' % (prefix[1], prefix[2]),
item).group(1)
        except AttributeError:
            continua
```

**MapReduce**

## Hive



## SQL

```
SELECT a.sscore, a.query, count(*)
FROM
  (SELECT sscore, query
   FROM korea_nx_mobile
   WHERE ....
  UNION ALL
  SELECT sscore, query
   FROM korea_nx_pc
   WHERE ....
  ) a
GROUP BY a.sscore, a.query
```

SQL

## Interfaces

- SCAN\_DUMP
- RANDOM\_DUMP
- MAP\_REDUCE
- STREAM\_DUMP
- STREAM\_API
- READER\_API
- WEB\_API
- HDFS
- SQL**

SQL 신청



# SQL Tool



```
$ beeline
0: lago@korea_nxlog_per_day [korea_real]> show databases;
-----+-----
| database_name |
+-----+-----
| korea_ai_public_dataset |
| korea_arts |
| korea_audioclip |
| korea_clova |
| korea_kin |
| korea_nxlog_per_day |
```

```
0: lago@korea_nxlog_per_day [korea_real]> SELECT query, count(*)
. . . . . > FROM nx_mobile
. . . . . > WHERE log_date='2019-10-10'
. . . . . > AND query='bts'
. . . . . > GROUP BY query;
```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	30	30	0	0	0	0
Reducer 2	container	SUCCEEDED	50	50	0	0	0	0

```
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 12.10 s
```

query	_c1
bts	18594

The screenshot shows the HUE web interface. On the left, there is a sidebar with a tree view of databases, including 'korea\_ai\_public\_dataset', 'korea\_arts', 'korea\_audioclip', 'korea\_clova', 'korea\_kin', and 'korea\_nxlog\_per\_day'. The main area contains a SQL query editor with a query that selects various columns from a table, including 'period', 'sscode', 'refine\_query', 'q\_sc', 'q\_cc', 'area', 'a\_sc', and 'a\_cc'. Below the query editor, there are several input fields for query parameters like 'PERIOD', 'START\_DATE', 'END\_DATE', 'SSCODE', 'UPLOAD\_QUERY\_FILE', 'QUERY\_MATCH', 'AREA', 'URL', 'U\_CC\_LIMIT', and 'U\_CC\_RANK'.

The screenshot shows the Zeppelin Notebook interface. At the top, it says 'Zeppelin Notebook Job'. Below that, the notebook name 'lago\_test' is visible. The main content area shows a code block with the command '%jdbc(lago) show databases;'. Below the code, there is a table with the output of the command, listing several database names: 'korea\_ai\_public\_dataset', 'korea\_amelie\_naver\_blog', 'korea\_arts', 'korea\_audioclip', 'korea\_clova', and 'korea\_cq'.







# 옛것에서 혁신을 찾다.



대한민국 법원  
종합법률정보

검색어를 입력하세요.

화제의판례 | 최신선고판례

검색결과 454건 20건

전체	NO	요약정보
<input type="checkbox"/> 선고구분 판결(436) 결정(18)	1	대법원 2019. 4. 18. 선고 2016다2451 전원합의체 판결 ★ [임금] <변경된 취업규칙상 소무효인지에 관한 사건> [공2019상,1074] <a href="#">요지 전체보기</a> <a href="#">화제</a> [다수의견] (가) 구 근로기준법(2018. 3. 20. 법률 제15513호로 개정되기 전의 것)은 휴게시간을, 1일의 근로시간은 8시간을 초과할 수 없도록 기준근로시간을 정하여 규제하면서 범위 내에서 근로자와 사용자가 합의한 근로시간을 소정근로시간으로 규정하고 있다( 제: 근로자는 합의한 소정근로시간 동안 근로의무를 부담하고, 사용자는 근로의무이행에 대해 근로자는 기준근로시간을 초과하지 않는 한 원칙적으로 자유로운 의사에 따라 소정근로시간의 정함이 단지 형식에 불과하다고 평가할 수
<input type="checkbox"/> 사건종류 민사(174) 형사(145) 가사(13) 특허(13) 세무(26) 일반행정(83)	2	대법원 2019. 4. 18. 선고 2017도14609 전원합의체 판결 ★ [마약류관리에관한법률위반(경합범)에 관한 감경을 할 때에 형기의 2분의 1 미만으로도 감경할 수 있는지 여부에 관한

# SQL 문서 전략 - Template

서식명	제목
로그데이터 권한신청서/업무...	[로그 분석의뢰] 급상승검색어를 통한 검색 통계 요청
로그데이터 권한신청서/업무...	[로그 분석의뢰] Cuve korea/nxlog_per_day/privacy 조회 권한 신청
로그데이터 권한신청서/업무...	[로그 분석의뢰]
로그데이터 권한신청서/업무...	[로그 분석의뢰] "빠른길찾기" 쿼리 검색 인입 질의에 대한 QC 기록
로그데이터 권한신청서/업무...	[로그 분석의뢰]
로그데이터 권한신청서/업무...	[로그 분석의뢰]
로그데이터 권한신청서/업무...	[로그 분석의뢰] 로그에 위치정보 포함여부 확인을 위한 로그추출 요청(LAIM-클릭로그 추출요청)
로그데이터 권한신청서/업무...	[로그 분석의뢰] 로그에 위치정보 포함여부 확인을 위한 로그추출 요청(LAIM)
로그데이터 권한신청서/업무...	[로그 분석의뢰]
로그데이터 권한신청서/업무...	[로그 분석의뢰] 매크로 통검 순위 조작 수사 대응 건 (인천지방청)

전수조사



노출수/클릭수/CTR

QC/UQC

노출문서

유입경로

연령/성별



# SQL 문서 전략 - Template

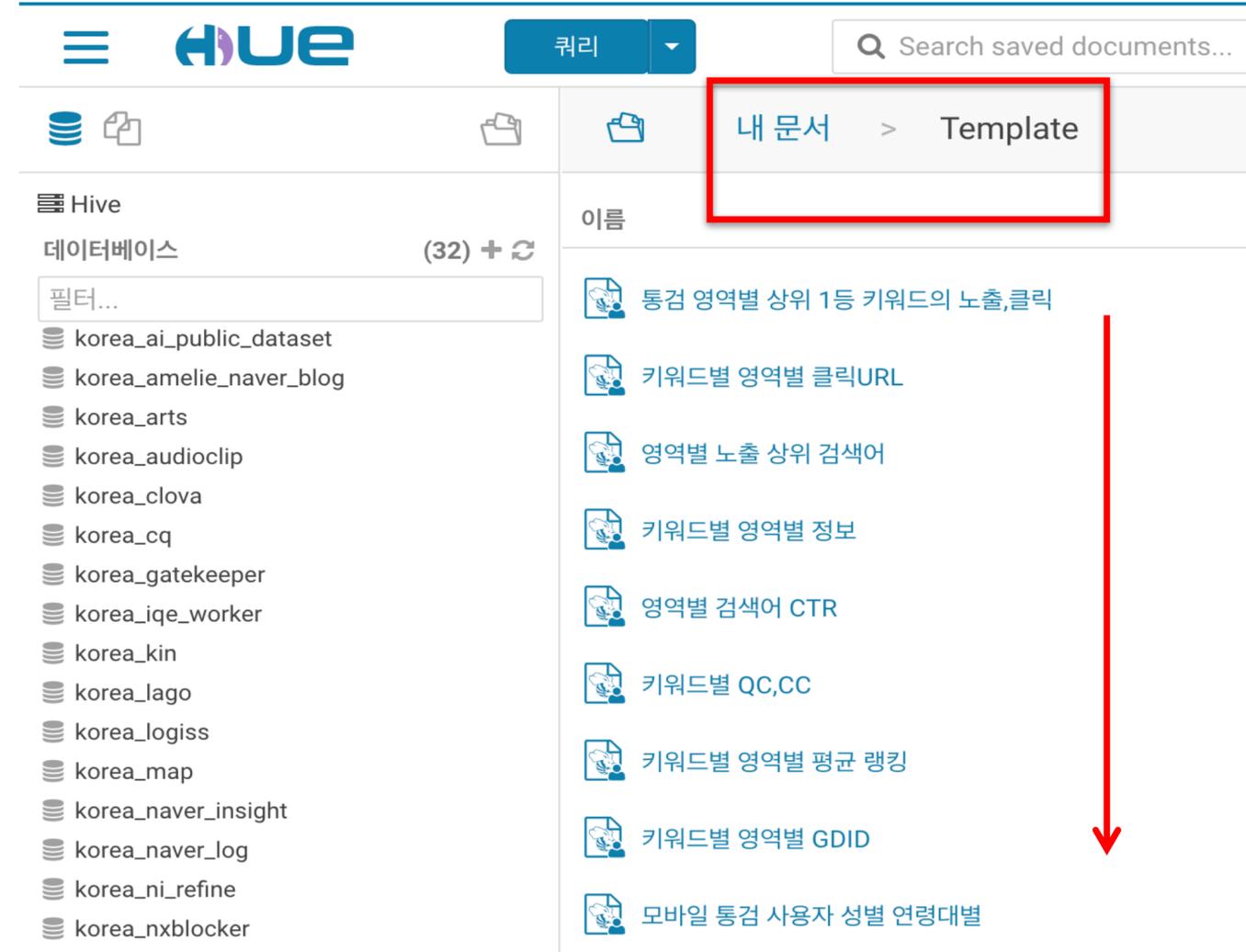
노출수/클릭수/CTR

QC/UQC

노출문서

유입경로

연령/성별



HUE

쿼리

Search saved documents...

내 문서 > Template

Hive  
데이터베이스 (32) + ↻

이름

- 통검 영역별 상위 1등 키워드의 노출,클릭
- 키워드별 영역별 클릭URL
- 영역별 노출 상위 검색어
- 키워드별 영역별 정보
- 영역별 검색어 CTR
- 키워드별 QC,CC
- 키워드별 영역별 평균 랭킹
- 키워드별 영역별 GDID
- 모바일 통검 사용자 성별 연령대별

# SQL 문서 전략 - Template



☰ HUE 쿼리

🔍 Search saved documents...

📄 내 문서 > Template

Hive 데이터베이스 (32) + ↻

필터...

- korea\_ai\_public\_dataset
- korea\_amelie\_naver\_blog
- korea\_arts
- korea\_audioclip
- korea\_clova
- korea\_cq
- korea\_gatekeeper
- korea\_iqe\_worker
- korea\_kin
- korea\_lago
- korea\_logiss
- korea\_map
- korea\_naver\_insight
- korea\_naver\_log
- korea\_ni\_refine
- korea\_nxblocker

이름

- 통검 영역별 상위 1등 키워드의 노출,클릭
- 키워드별 영역별 클릭URL
- 영역별 노출 상위 검색어
- 키워드별 영역별 정보
- 영역별 검색어 CTR
- 키워드별 QC,CC
- 키워드별 영역별 평균 랭킹
- 키워드별 영역별 GDID
- 모바일 통검 사용자 성별 연령대별

# SQL 문서 전략 - Template

```
12      nx_cr.sc area_sc,  
13      nx_cr.cc area_cc  
14 FROM  
15      -- Q in ('네이버') | SC, CC  
16      -- 쿼리에 대한 노출/클릭 추출  
17      ( SELECT lago.to_period(log_date, '${PERIOD=all(전체기간)}, year(연도별), half(반기별), quarter(분기별), mont  
18          refine_query,  
19          sum(sc) sc,
```

PERIOD 전체기간 START\_DATE 2019. 10. 04. END\_DATE 2019. 07. 15.

SSCODE 모바일통검 QUERY 네이버

2019년 10월

일	월	화	수	목	금	토
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

이름      소유자

# SQL 문서 전략 - Template



```
17 ( SELECT lgo.to_period(log_date, '${PERIOD=all(전체기간)}, year(연도별), half(반기별), quarter(분기별), month(월별)')
18     , refine_query,
19     sum(sc) sc,
20     sum(cc) cc
21 FROM korea_nxlog_per_day.ctr_query
22 WHERE log_date >= '${START_DATE}'
23     AND log_date <= '${END_DATE}'
24     AND sscore='${SSCODE=tab.m.all(모바일통검), tab.nx.all(PC통검)}'
```

PERIOD	전체기간	START_DATE	2019. 10. 01.	END_DATE	2019. 10. 28.
SSCODE	모바일통검	QUERY	방탄소년단	COVERAGE	major

	period	query	sc	cc	area	nx_area	cr_area	area_sc	area_cc
22	Whole period	방탄소년단	2118354	2817961	rsk	rsk	rsk	2118353	90580
7	Whole period	방탄소년단	2118354	2817961	gnb	-	gnb	0	79643
6	Whole period	방탄소년단	2118354	2817961	fsn	fsn	fsn	2118352	7439

# SQL 문서 전략 – SQL-Archive



- 🔔** 검색\_게임 키워드 동영상 콘텐츠 QC, 노출수, 클릭수 **ClickCount(CC)** **QueryCount(QC)** **검색노출Count(SC)** **통검로그**  
#39 by soonho-hwang was closed on 4 Jul
- 🔔** 특정기간의 통검 컬렉션 성/연령별 클릭사용자 **ClickCount(CC)** **통검로그**  
#38 by youngho-na was closed on 4 Jul
- 🔔** 뮤직 곡 정답형 기간별 유니크 검색 사용자 파악 **검색노출Count(SC)** **유니크 사용자**  
#37 by jh-r was closed on 25 Jun 0 of 2
- 🔔** 모바일 통합검색 연령대 별 상위 검색어 **검색노출Count(SC)** **연령별** **완료**  
#36 by jeeun-kim was closed on 20 Jun
- 🔔** 통검 컬렉션의 주간/월간 클릭사용자 **영역** **완료** **유니크 사용자** **이번주** **통검로그**  
#35 by youngho-na was closed on 18 Jun

사내 github

# SQL 문서 전략 – SQL-Archive



**🔔** 검색\_ 게임 키워드 동영상 콘텐츠 QC, 노출수, 클릭수 **ClickCount(CC)** **QueryC**  
#39 by soonho-hwang was closed on 4 Jul

**🔔** 특정기간의 통검 컬렉션 성/  
#38 by youngho-na was closec



쿼리

Search saved documents...



내 문서 > sql\_archive

모두



**🔔** 뮤직 곡 정답형 기간별 유니  
#37 by jh-r was closed on 25 J

이름

설명

유형

소유자



issues\_#39

https://oss.navercorp.com/lago/sql-archive/is...

Hive 쿼리

admin

**🔔** 모바일 통합검색 연령대 별  
#36 by jeeun-kim was closed o



issues\_#29-6

https://oss.navercorp.com/lago/sql-archive/is...

Hive 쿼리

admin



issues\_#29-5

https://oss.navercorp.com/lago/sql-archive/is...

Hive 쿼리

admin

**🔔** 통검 컬렉션의 주간/월간 클  
#35 by youngho-na was closec



issues\_#29-4

https://oss.navercorp.com/lago/sql-archive/is...

Hive 쿼리

admin



issues\_#29-3

https://oss.navercorp.com/lago/sql-archive/is...

Hive 쿼리

admin

# SQL 문서 전략 – SQL-Archive



쿼리

Search saved documents...

Job

Hive

issues\_#39

https://oss.navercorp.com/lago/sql-archive/...



default

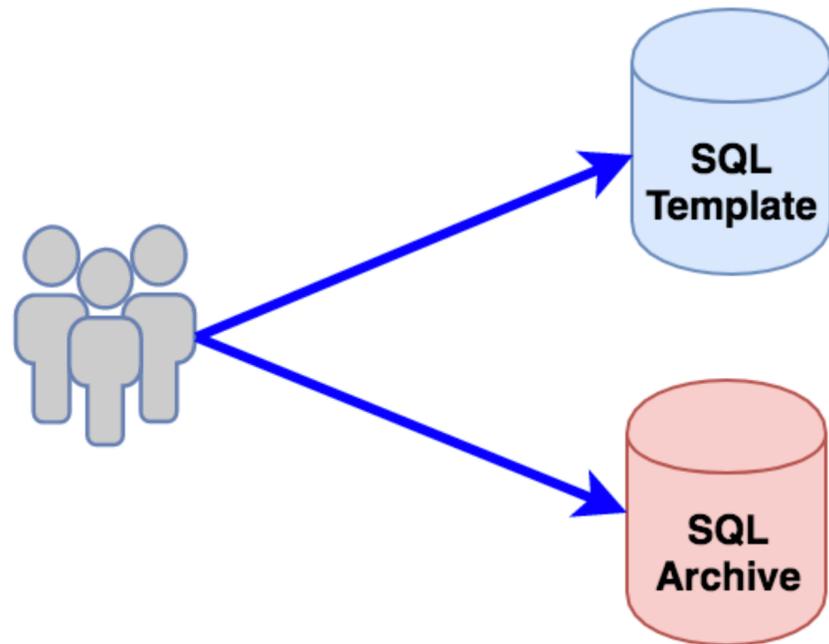
```
31 AND url LIKE ('${URL=%tv.naver%, %youtube%}')
32 AND area LIKE 'vdo%'
33 GROUP BY sscore,
34     2,
35     refine_query ) AS t_url ON (t_ctr.refine_query = t_url.refine_query
36     AND t_ctr.sscore = t_url.sscore
37     AND t_ctr.period = t_url.period)
```

PERIOD	분기별	START_DATE	2019. 06. 30.	END_DATE	2019. 10. 29.
SSCODE	연도별	MATCH	정확히 일치	URL	%tv.naver%
	반기별				
	분기별				
쿼리 기록	월별				

쿼리 빌더

# 로그 분석은 누구라도~

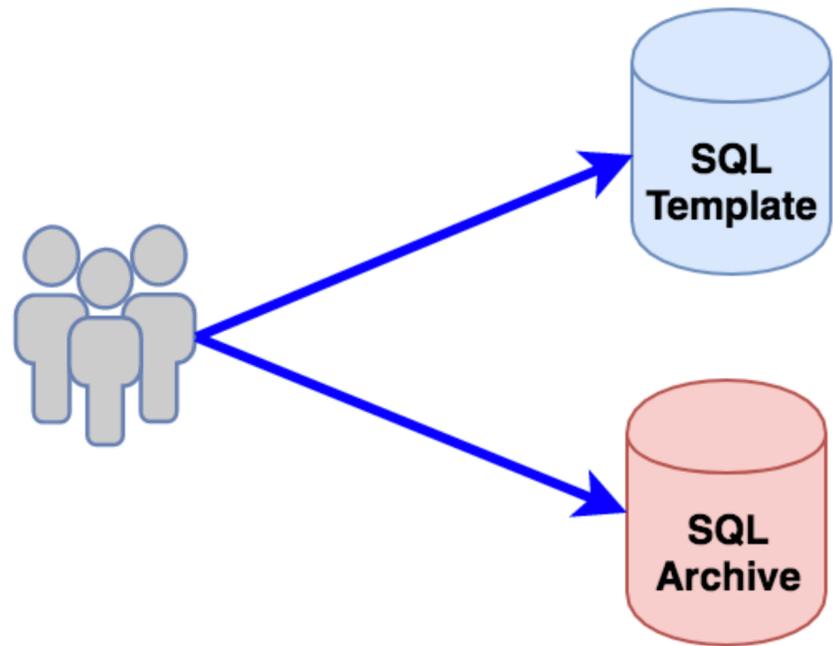
## 유사 분석의뢰 찾기



내 문서 > Template	
이름	설명
통검 영역별 상위 1등 키워드의 노출,클릭	버전 : 2019-07-31 문서 : <a href="https://pages.oss.nave...">https://pages.oss.nave...</a>
키워드별 영역별 클릭URL	버전 : 2019-01-09 문서 : <a href="https://oss.navercorp.c...">https://oss.navercorp.c...</a>
영역별 노출 상위 검색어	버전 : 2019-01-09 문서 : <a href="https://oss.navercorp.c...">https://oss.navercorp.c...</a>

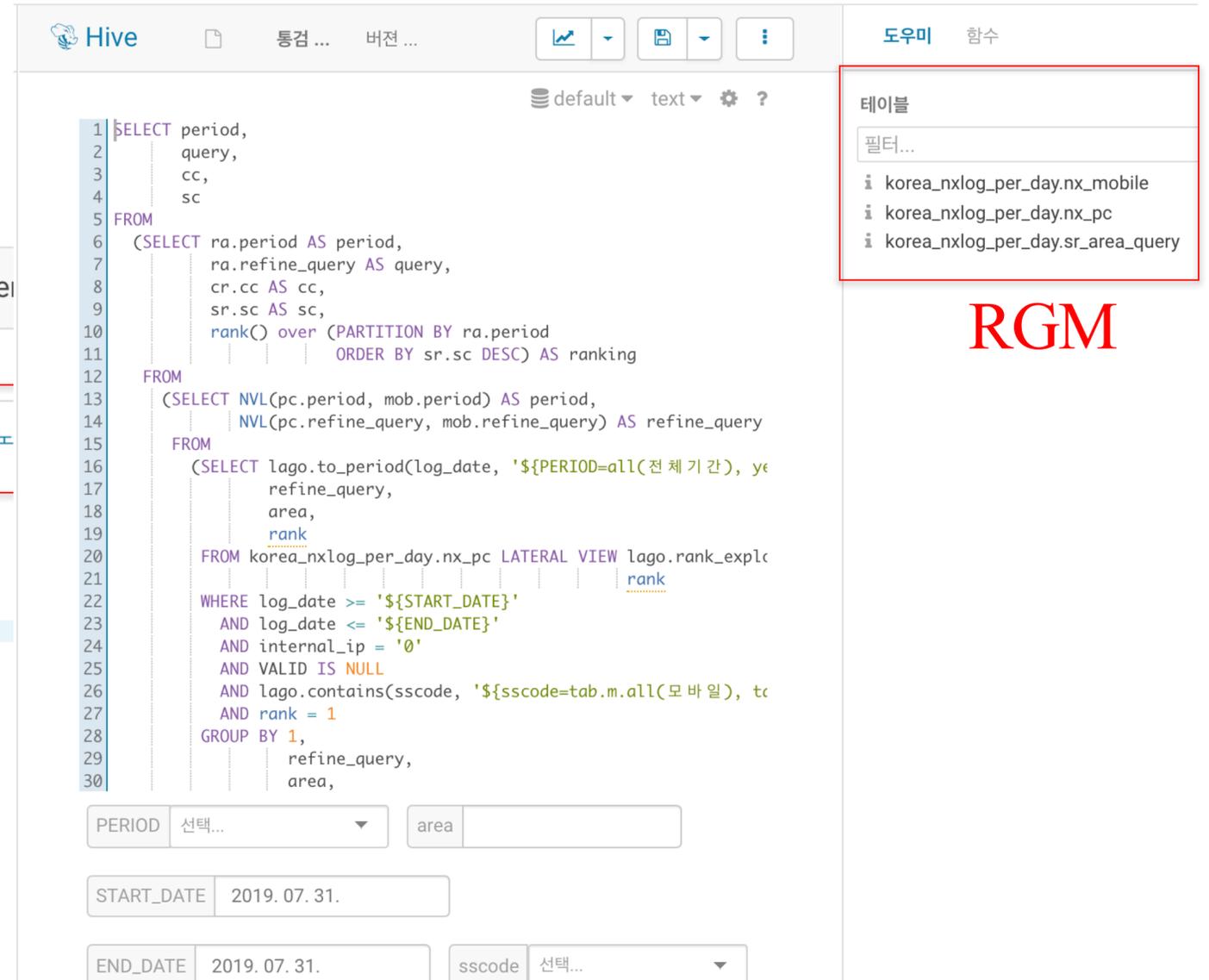
# 로그 분석은 누구라도~

## 유사 분석의뢰 찾기



내 문서 > Te

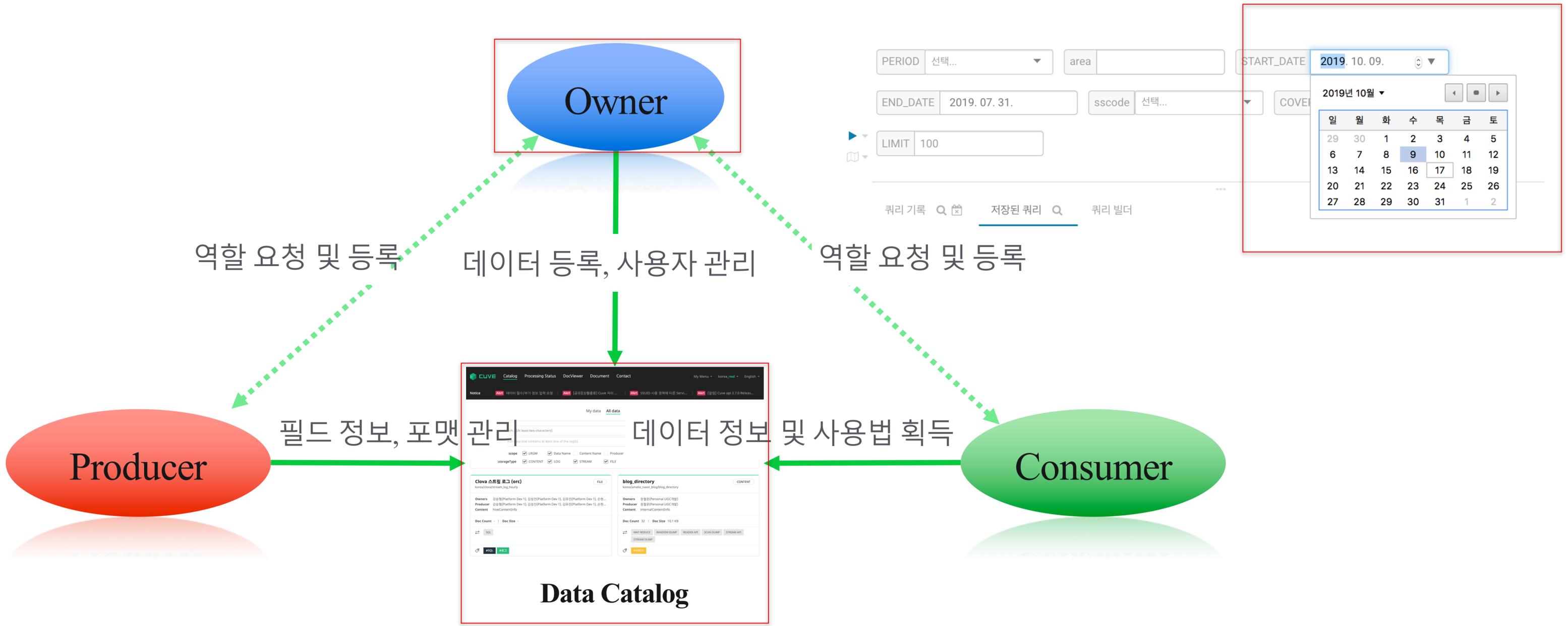
이름
통검 영역별 상위 1등 키워드의 노
키워드별 영역별 클릭URL
영역별 노출 상위 검색어



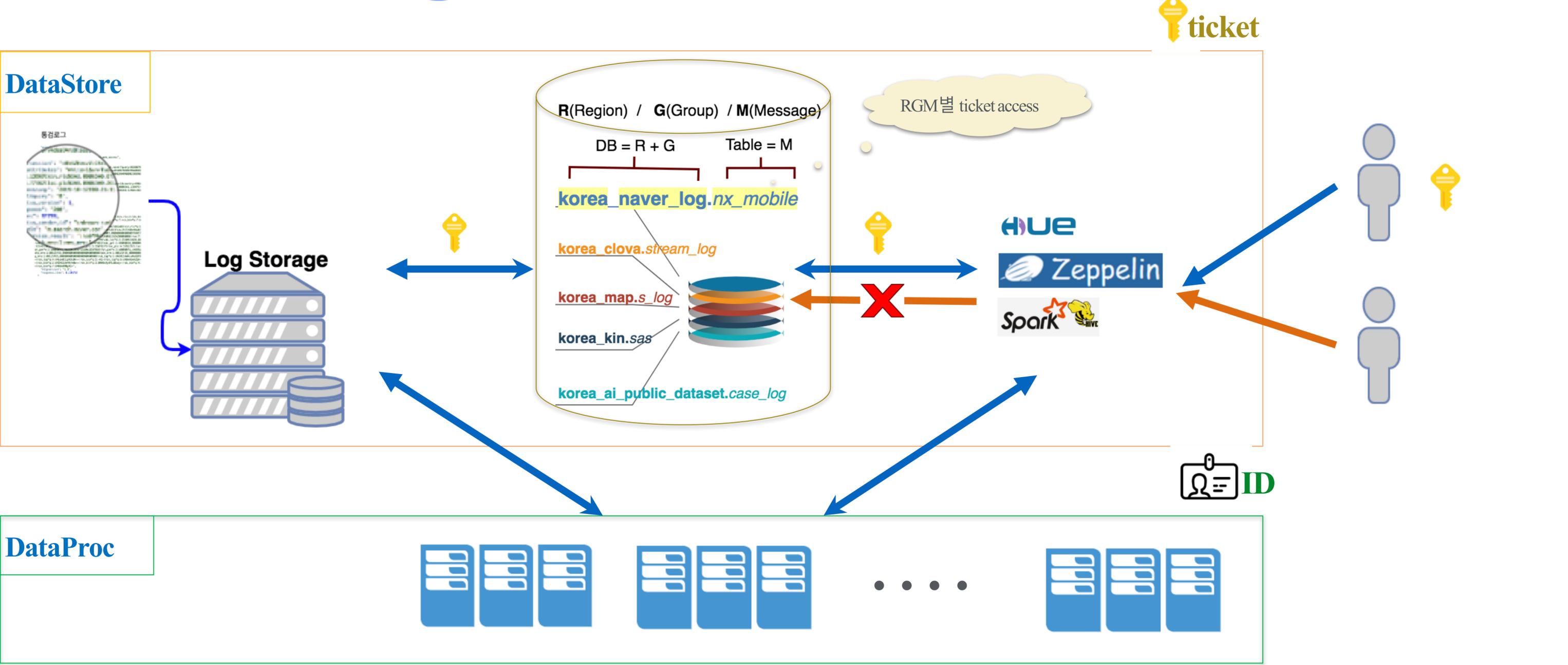
The screenshot shows the Hive SQL editor interface. The main area contains a complex SQL query for finding top keywords by region. The query includes sub-queries for ranking keywords and joining with a log table. Below the query, there are input fields for 'PERIOD', 'START\_DATE' (2019. 07. 31.), 'END\_DATE' (2019. 07. 31.), and 'sscode'. On the right side, a sidebar titled '테이블' (Table) shows a filter for 'korea\_nxlog\_per\_day.nx\_mobile'. Below the sidebar, the text 'RGM' is displayed in red.

```
1 | SELECT period,
2 |       query,
3 |       cc,
4 |       sc
5 | FROM
6 | (SELECT ra.period AS period,
7 |       ra.refine_query AS query,
8 |       cr.cc AS cc,
9 |       sr.sc AS sc,
10 |      rank() over (PARTITION BY ra.period
11 |                  ORDER BY sr.sc DESC) AS ranking
12 | FROM
13 | (SELECT NVL(pc.period, mob.period) AS period,
14 |       NVL(pc.refine_query, mob.refine_query) AS refine_query
15 | FROM
16 | (SELECT lago.to_period(log_date, '${PERIOD=all(전체기간)}, ye
17 |       refine_query,
18 |       area,
19 |       rank
20 | FROM korea_nxlog_per_day.nx_pc LATERAL VIEW lago.rank_explc
21 |      rank
22 | WHERE log_date >= '${START_DATE}'
23 |       AND log_date <= '${END_DATE}'
24 |       AND internal_ip = '0'
25 |       AND VALID IS NULL
26 |       AND lago.contains(sscode, '${sscode=tab.m.all(모바일)}, tc
27 |       AND rank = 1
28 | GROUP BY 1,
29 |         refine_query,
30 |         area,
```

# 로그 분석은 누구라도~

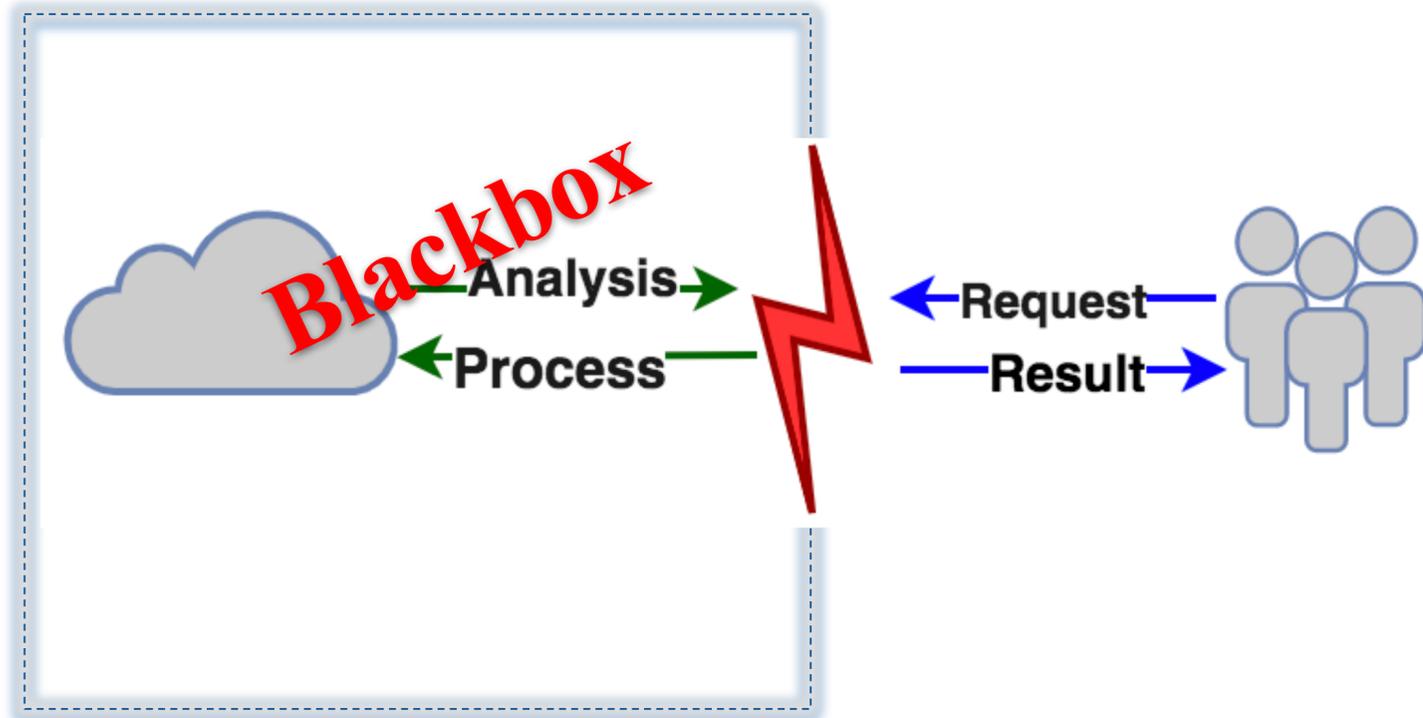


# DataStore 인증

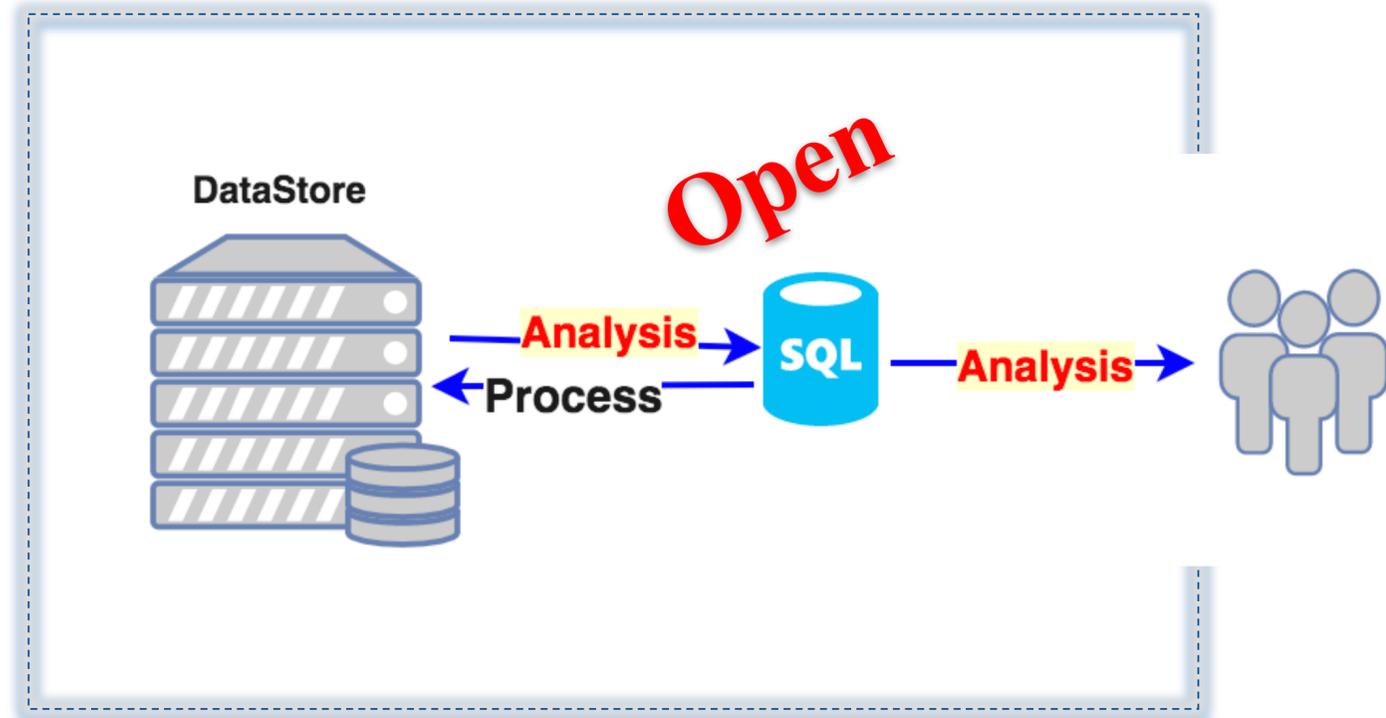


# 오픈형 로그 분석

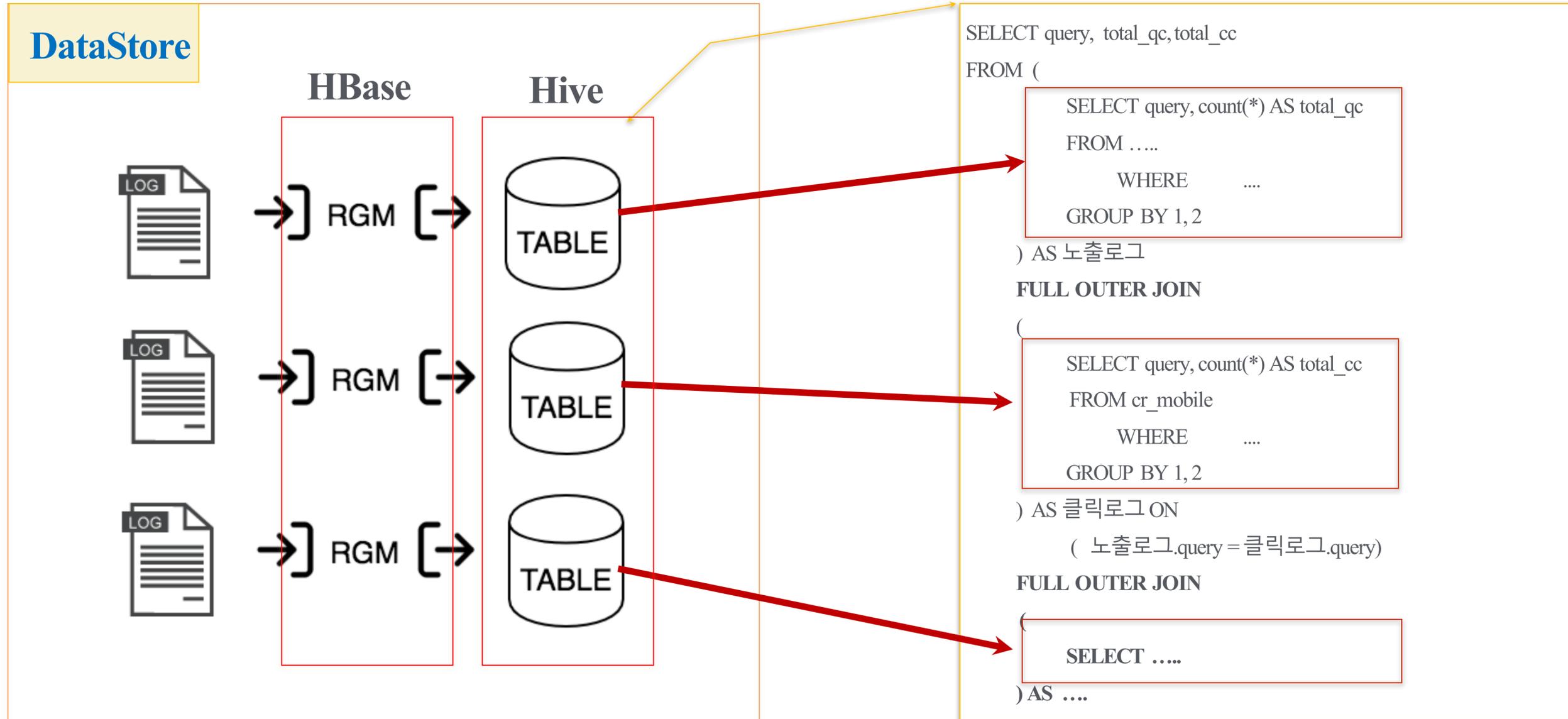
before



after

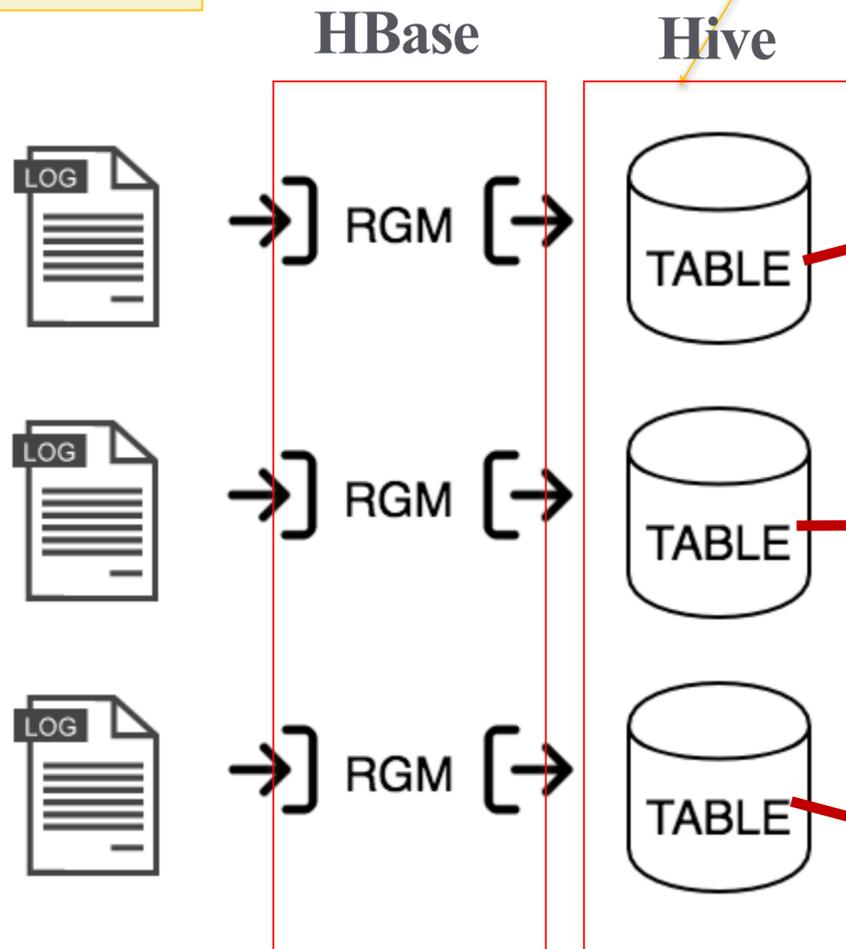


# 이게 최선입니까?



# 이게 최선입니까?

## DataStore



```
SELECT query, total_qc, total_cc  
FROM (  
  SELECT query, total_qc  
  FROM .....  
  WHERE .....  
  GROUP BY 1, 2  
) AS 노출로그  
FULL OUTER JOIN  
(  
  SELECT query, .....  
  FROM .....  
  WHERE .....  
  GROUP BY 1, 2  
) AS ..... ON  
  (노출로그.query = 클릭로그.query)  
FULL OUTER JOIN  
(  
  SELECT .....  
) AS .....
```

추출 시 복잡한 집계 연산

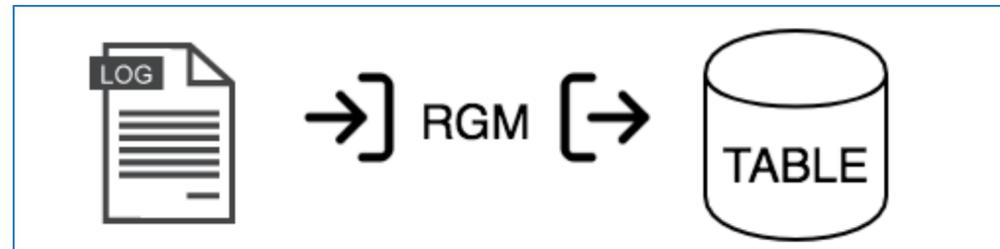
여러 테이블 JOIN

SQL 너무 복잡하고 어려워요.  
어떤 테이블을 사용해야 되죠?  
추출이 너무 오래 걸려요.



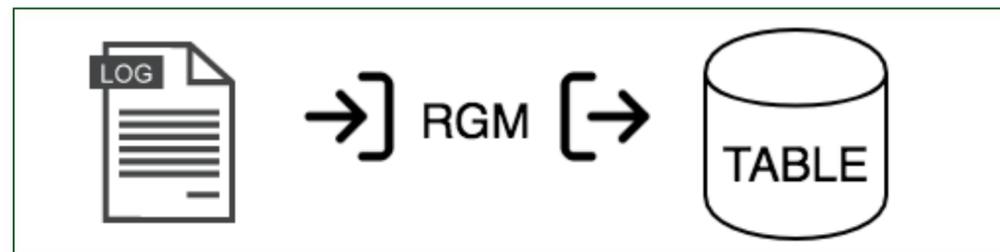
# 이게 최선입니까?

## DataStore



### 노출로그

```
SELECT query, count(*)  
FROM ....
```



### 클릭로그

```
SELECT query, count(*)  
FROM ...
```

join



### 검색어 별 노출수, 클릭수 집계 Table

검색어	노출수	클릭수
방탄소년단	79577	111258
엑소	8240	12847
갯세븐	2259	3178



batch

# 원본 Table vs 가공 Table

DataStore

원본로그 TABLE

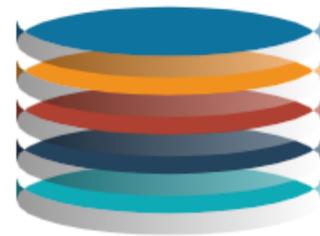
노출로그

클릭로그

검색문서정보

컬렉션정보

지표정보

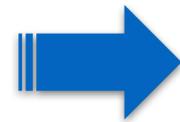
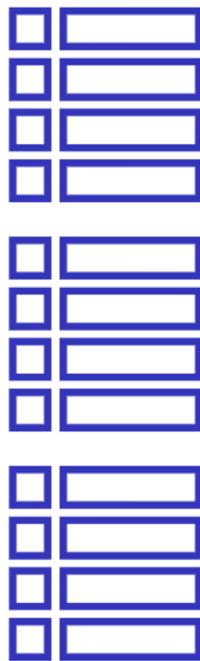


가공 Table

쿼리별 영역에 대한 클릭 집계 Table

쿼리별 CTR(노출대비 클릭) 집계 Table

영역별 노출 문서 집계 Table

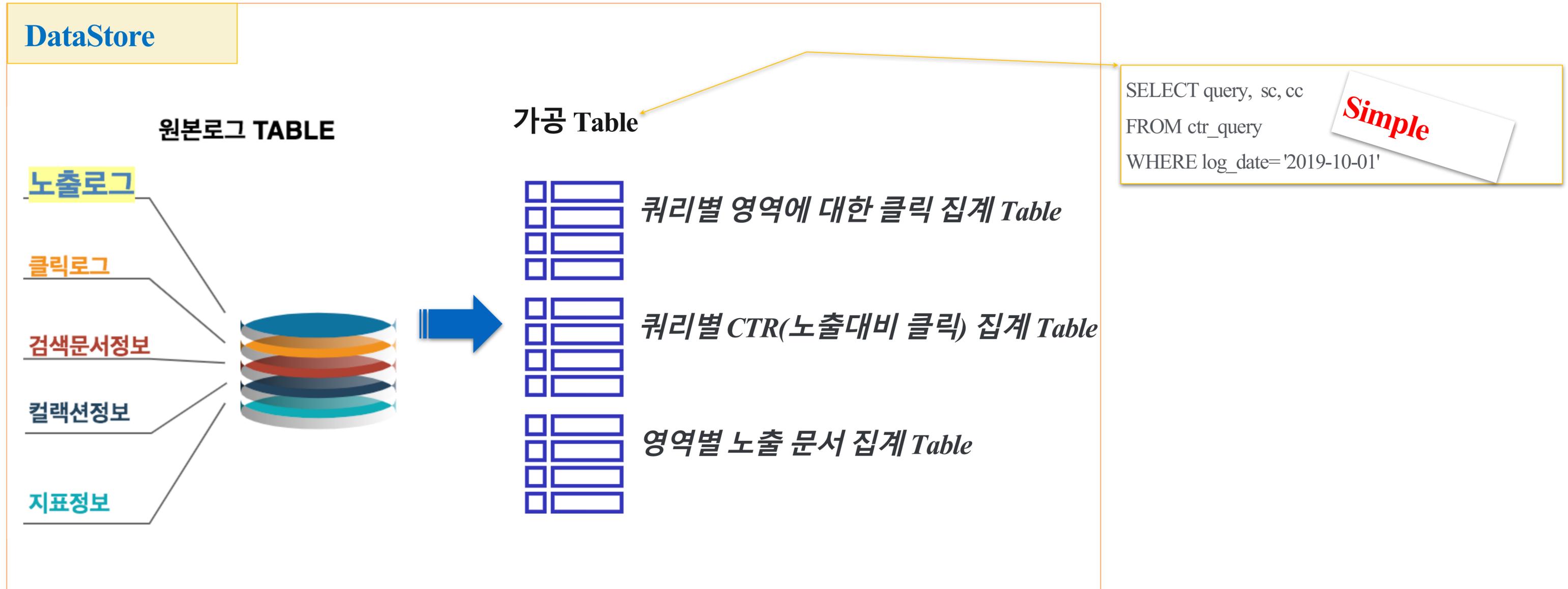


```

SELECT query, total_qc, total_cc
FROM (
    SELECT query, count(*) AS total_qc
    FROM nx_mobile
    WHERE ...
    GROUP BY ...
) AS 노출
FULL OUTER JOIN
(
    SELECT query, count(*) AS total_cc
    FROM cr_mobile
    WHERE ...
    GROUP BY 1, 2
) AS 클릭로그 ON
( 노출로그.query = 클릭로그.query)
    
```

**복잡한 SQL**

# 원본 Table vs 가공 Table





# 가공 Table 성능 (ORC Format)

하루 치 모바일 통합검색 특정 2개 컬럼만 추출  
SELECT query, sscore FROM ...

## 원본 Table

Details <b>Hbase</b>		Stats	
<a href="#">Download data</a>		Total Vertices	1
Application ID	application_1559206308011_24845492	Succeeded Vertices	1
ID	dag_1559206308011_24845492_1	Total Tasks	328
Name	SELECT query, sscore ...og_date='2019-09-21'(Stage-1)	Succeeded Tasks	328
Submitter	nowol	Failed Tasks	0
Status	✓ SUCCEEDED	Killed Tasks	0
Progress	100%	Failed Task Attempts	0
Start Time	24 Sep 2019 13:04:42	Killed Task Attempts	0
End Time	24 Sep 2019 13:32:04	HDFS_BYTES_WRITTEN	17.7G 19085791800
Duration	27m 22s 905ms	HDFS_READ_OPS	656
Queue	lago	HDFS_WRITE_OPS	656
Loas	1	PHYSICAL_MEMORY_BYTES	316G 340204716032
		VIRTUAL_MEMORY_BYTES	3601G 3866687549440

## 가공 Table

Details <b>ORC</b>		Stats	
<a href="#">Download data</a>		Total Vertices	1
Application ID	application_1559206308011_24845726	Succeeded Vertices	1
ID	dag_1559206308011_24845726_1	Total Tasks	192
Name	select query, sscore ...og_date='2019-09-21'(Stage-1)	Succeeded Tasks	192
Submitter	nowol	Failed Tasks	0
Status	✓ SUCCEEDED	Killed Tasks	0
Progress	100%	Failed Task Attempts	0
Start Time	24 Sep 2019 13:05:54	Killed Task Attempts	0
End Time	24 Sep 2019 13:06:21	HDFS_BYTES_WRITTEN	17.7G 19085771448
Duration	27s 347ms	HDFS_READ_OPS	768
Queue	lago	PHYSICAL_MEMORY_BYTES	118G 126672175104
		VIRTUAL_MEMORY_BYTES	2100G 2254899118080

# 가공 Table 성능 – PARTITONS

## 로그 분석 패턴

**매달** 십대가 검색한 '~뜻' 검색어...

**6/1~9/1일**까지 검색 상위 ...

**3월부터 현재까지** 시간/연령/성별 ...

**특정 기간**을 기준으로 로그 추출이 이루어짐

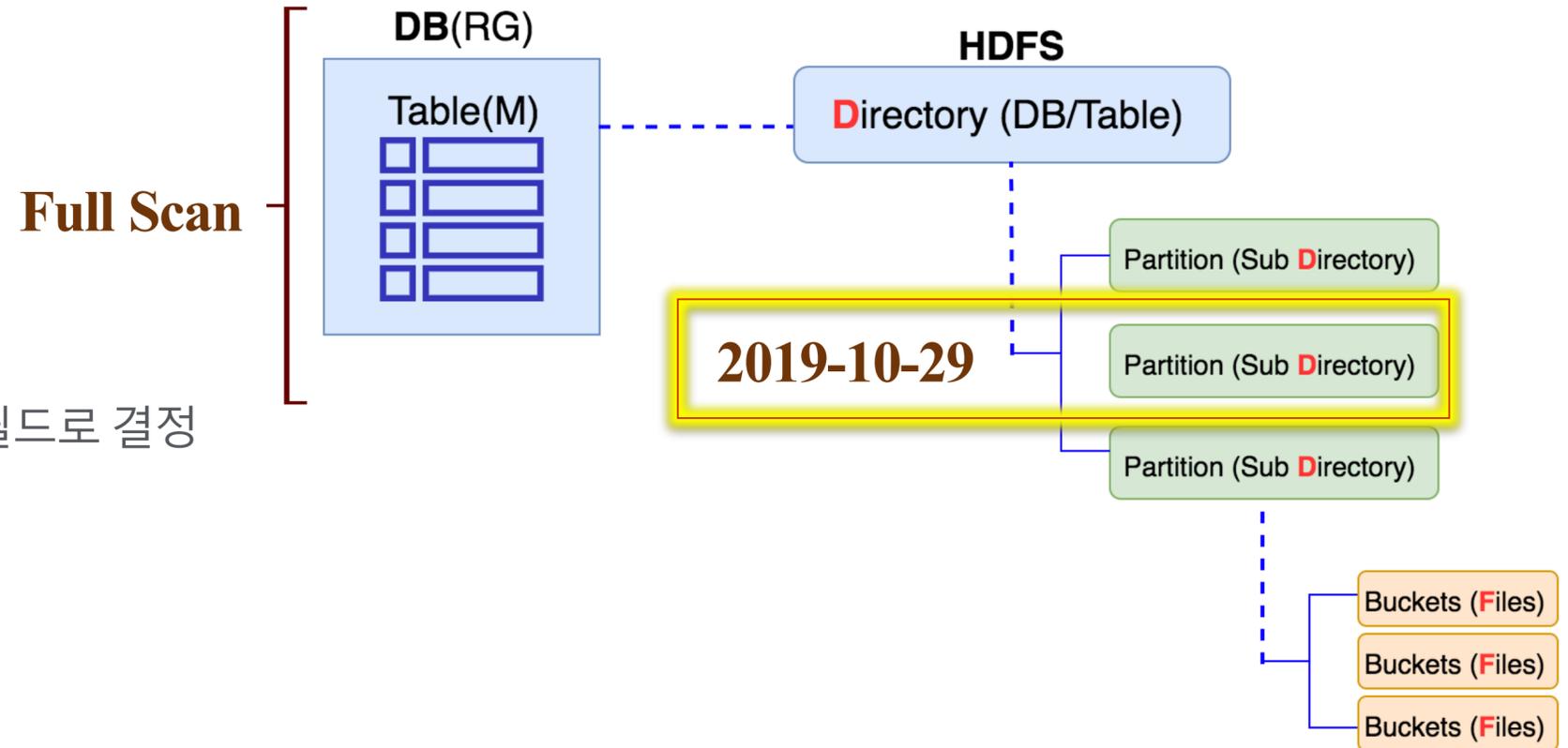
필요한 **날짜의 로그만 빠르게 조회**하기 위해서 partition은 날짜 필드로 결정

**PARTITIONED BY (log\_date date)**

SELECT \*

FROM ...

WHERE log\_date='2019-10-29' (파티션을 지정해 SELECT)



# 가공 Table 성능 - Buckets

## **CLUSTERED BY (컬럼) INTO 버킷개수 BUCKETS;**

컬럼을 해시 기준으로 지정된 버킷 개수의 파일로 분리해서 저장

버킷 개수 = File 수

정렬과 함께 사용하며,

## **join키로 bucket을 생성해 두면 파일 전체를 full scan하지 않음 필요한 파일만 사용**

매달 십대가 검색한 '~뜻' **검색어** 중 사전 컬렉션 영역 클릭률  
2018년 한 해 이미지 탭검 **상위 검색어**/탭검 진입경로/썸네일.....  
**검색어별** 컬렉션 랭킹 지표

검색어를 Bucket의 cluster 컬럼으로 사용함

**Full Scan**

**HDFS**

Directory (DB/Table)

Partition (Sub Directory)

Partition (Sub Directory)

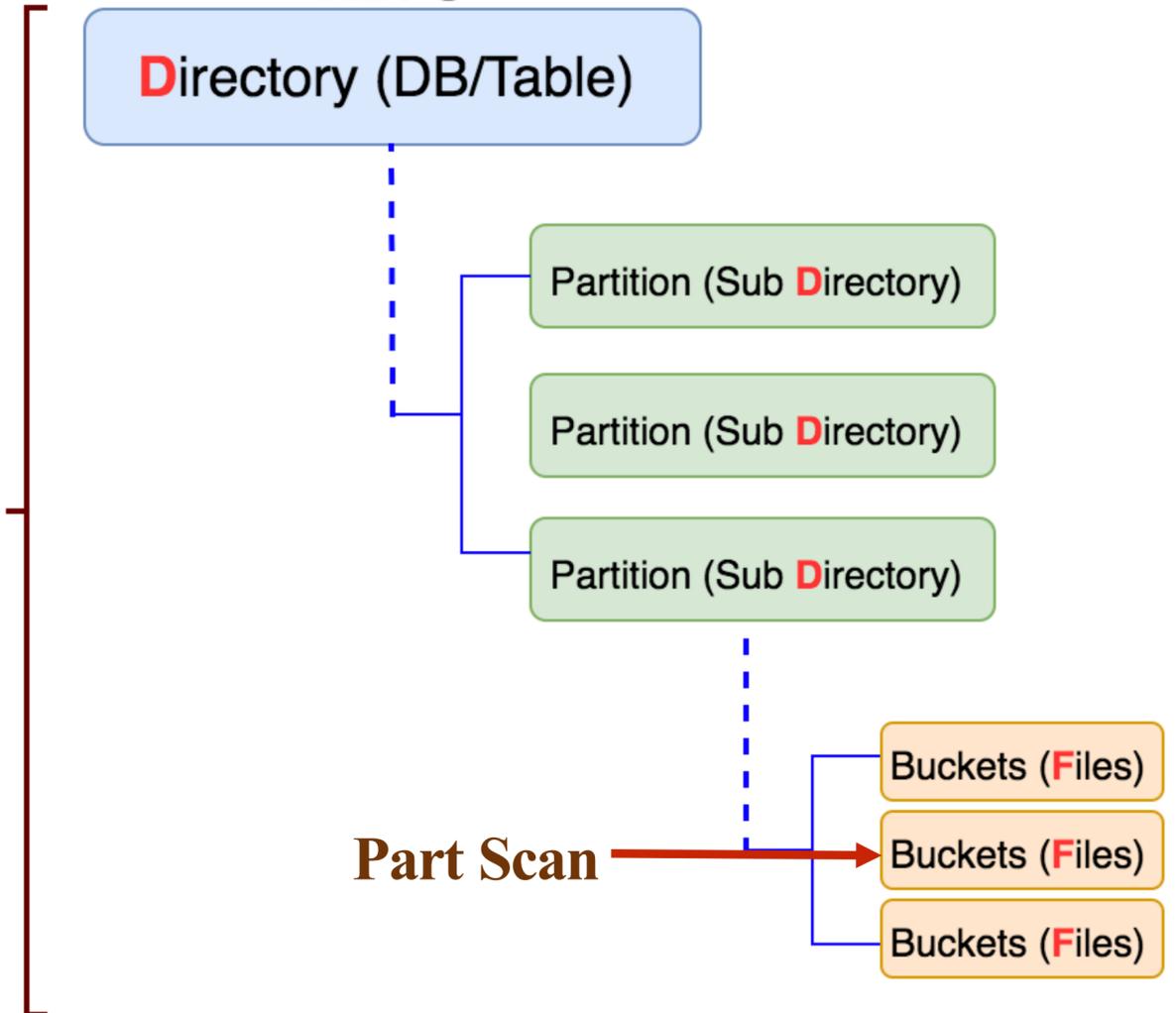
Partition (Sub Directory)

**Part Scan**

Buckets (Files)

Buckets (Files)

Buckets (Files)

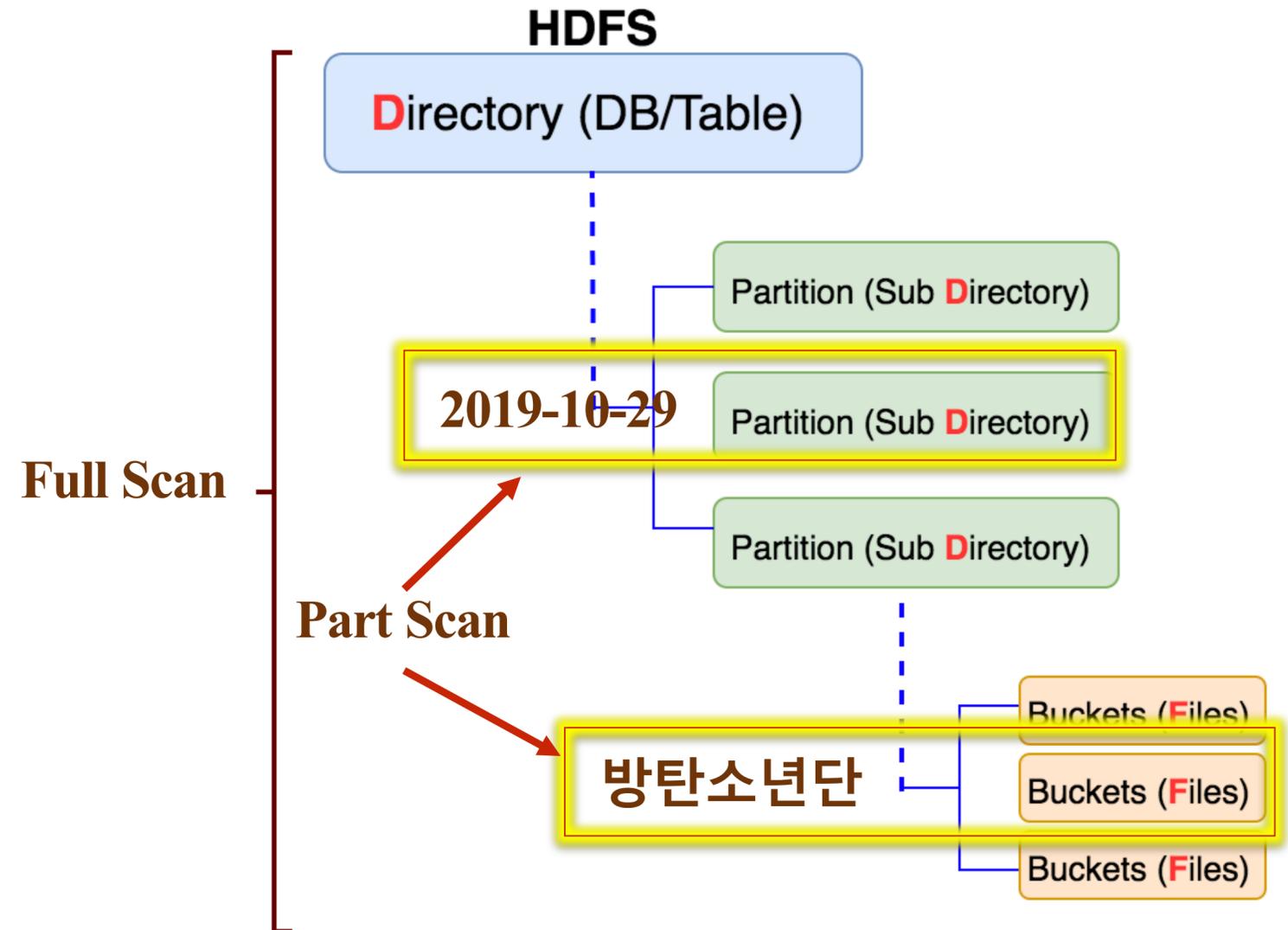


# 가공 Table 성능 - Buckets

Partition(**날짜**)별 = 디렉토리

Bucket(**검색어**) = 파일

```
SELECT ...  
FROM ...  
WHERE log_date='2019-10-29' ( 2019-10-29 디렉토리 하위 )  
AND query="방탄소년단" (방탄소년단이 포함된 파일만 Scan)
```



# 가공 Table 성능 - Buckets

검색 서비스인만큼, 검색어를 cluster 값으로 지정할 경우 주의가 필요하다.

연예인 극단적선택  
정치적/사회적 이슈  
지진, 태풍 등 천재지변  
등등

**특정 검색어가 집중되면,**

- 해당 검색어 bucket 파일 사이즈가 커짐
- 해당 파티션 해당 bucket의 생성 시간이 오래걸림
- shuffle 단계에서 메모리 에러가 발생 (OOM)

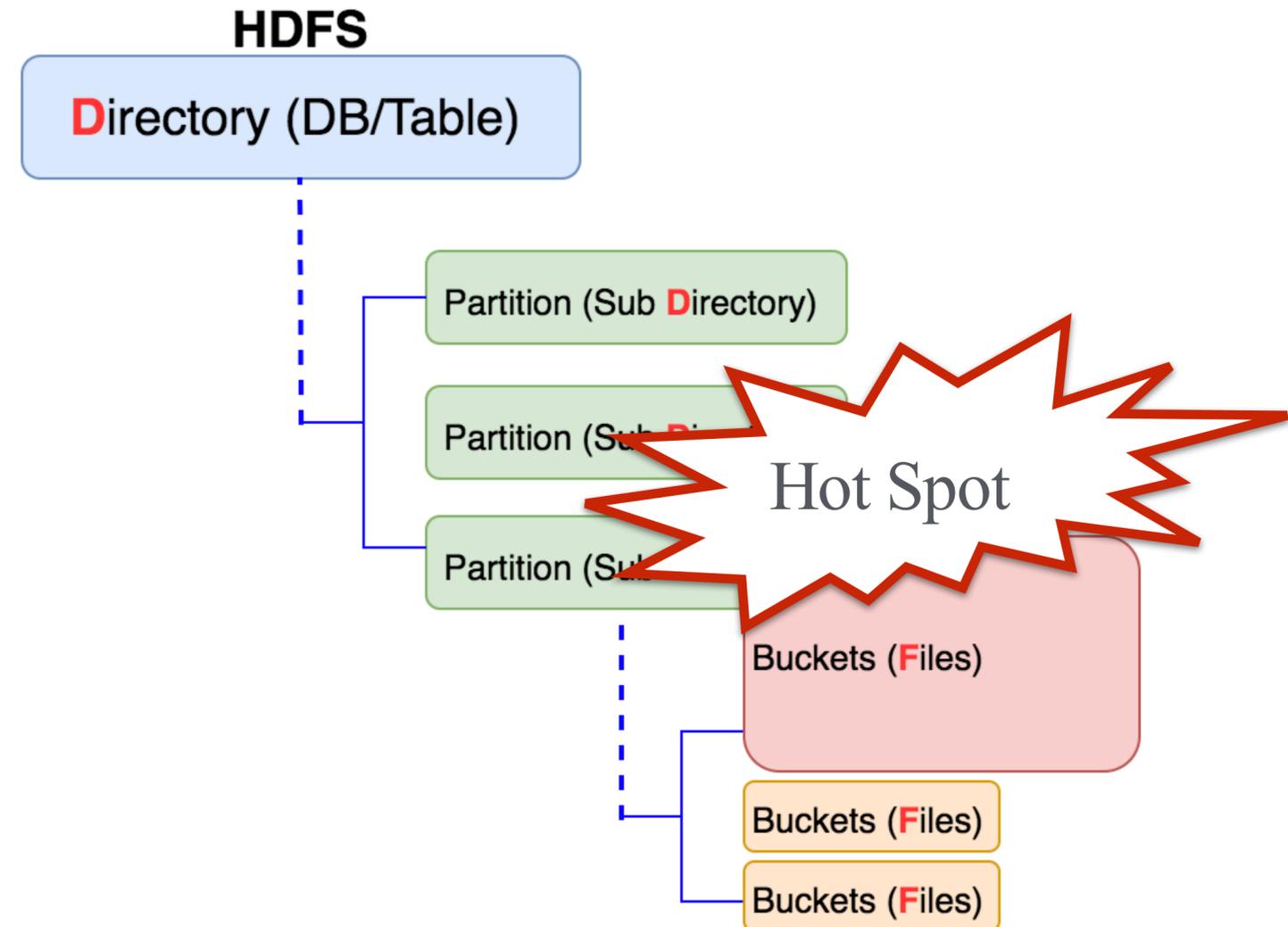
메모리를 늘려줌

*tez.am.resource.memory.mb*

DISK I/O를 감수하더라도 메모리 의존도를 내려줌

*tez.runtime.shuffle.fetch.buffer.percent*

*tez.runtime.shuffle.memory.limit.percent*



# 가공 Table 성능

원본 로그의 경우 HBase에 저장되기 때문에 External Table 이 필수  
가공 Table도 대부분 External Table을 사용

- External Table 이점

- 1. 안전하다.

- Table을 drop하더라도 실제 데이터가 지워지지 않는다.

- 2. HDFS가 아니라도 Table화 가능하다.

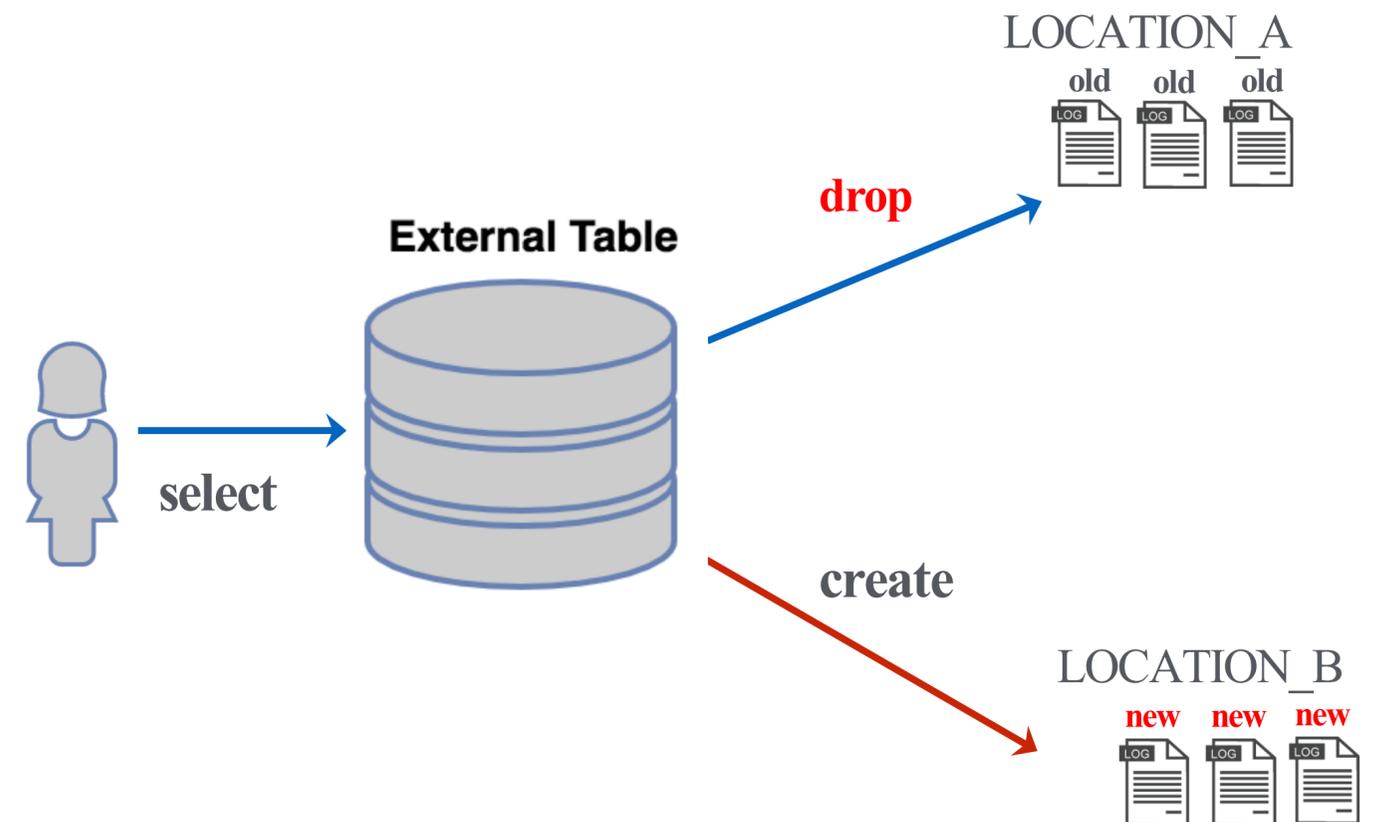
- Hbase, Cassandra, Azure Table, JDBC, ElasticSearch, Kafka...*

- 3. 데이터 소스 변경이 용이하다.

- Location지정으로 데이터 소스를 필요한 시점에 변경해 사용할 수 있다.

But.

*Hive에서 제공하는 transaction 이나 LLAP(Live Long And Process) 사용할 때  
제약이 있음*





# 그래서 지금은...

네이버 검색에서 소중한 로그는...

**DataStore 저장률 100%**

 Security 정보도 유효기간 이후 삭제

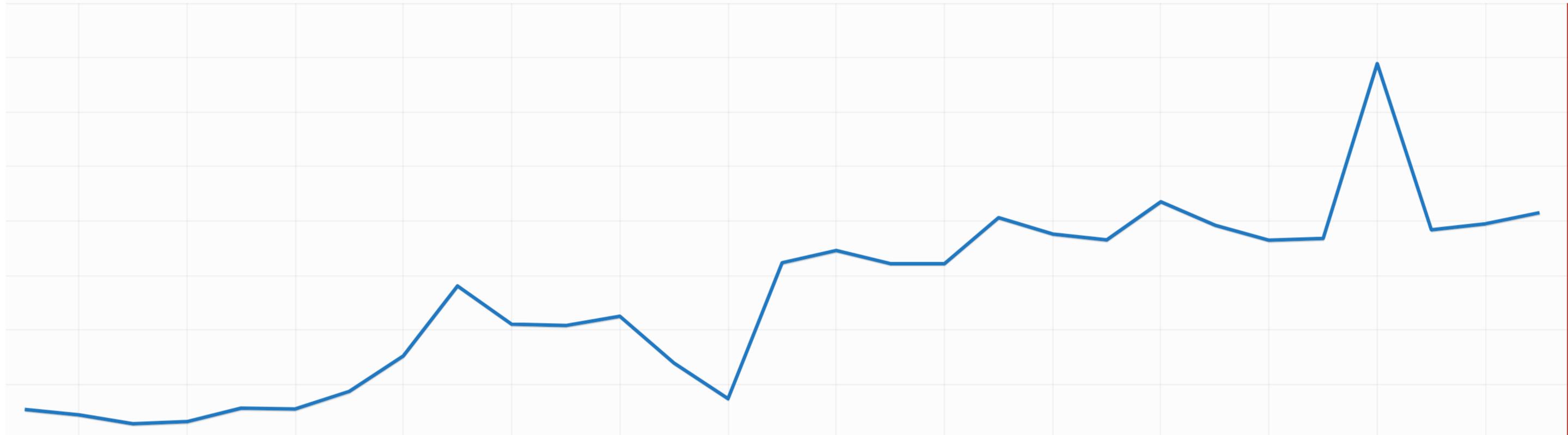
**Table 전환률 100%**

**SQL Interface 100%**

# 그래서 지금은...

우상향으로 순조롭게 순항중...

일간 SQL 실행횟수



# 앞으로..

## 표준화

- ✓ 사내 범용 로그저장소
- ✓ SQL Interface 확장

## 글로벌

- ✓ 글로벌 검색 로그의 표준
- ✓ 검색 로그 활용 일원화

## 성능

- ✓ LLAP & Presto 도입으로 쿼리 속도 증가
- ✓ Materialized views 도입 (SQL 단순화)
- ✓ Kafka storagehandler 등을 도입, 실시간 로그 활용

## 안정화

- ✓ Hive 3.x upgrade
- ✓ Secure Hadoop cluster 도입
- ✓ Oozie or airflow 도입 추출 작업 자동화

DataStore

# Q&A

DEVIEW  
2019



Big Data&AI Platform



Data Suite



AI Suite



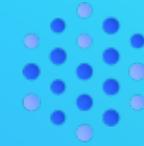
DataStore



DataProc



DataLog



AiFeatures



AiTraining



AiServing

