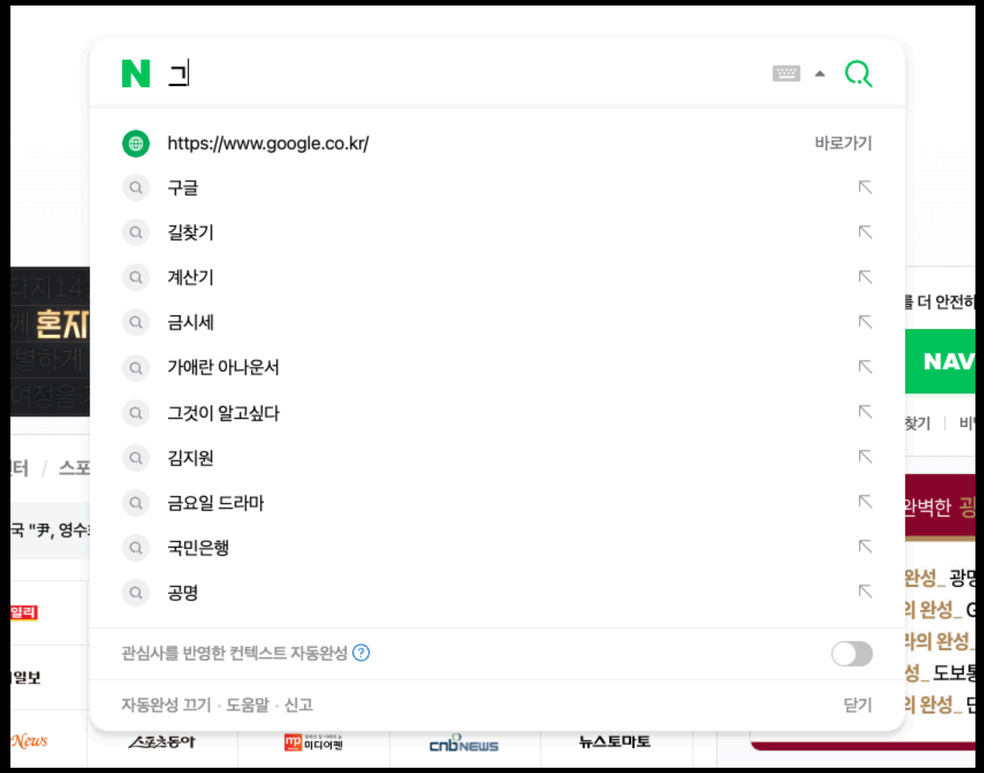
13장. 검색어 자동 완성 시스템

발표자: 제이(소재훈)

목차

1. 서론

사용자들이 많이 검색한 글자들을 보여주는 “검색어 자동 완성” 기능



2. 문제 이해 및 설계 범위 확정

- 면접관과의 대화를 통해 설계에 대한 요구사항 및 설계 범위를 확정함

👩‍🎓 지원자: 사용자가 입력하는 단어에서 자동완성될 검색어의 부분은 첫 부분인가요? 중간 부분인가요?

👩‍🎓 지원자: 몇 개의 자동 완성 검색어가 표시되어야 하나요?

👩‍🎓 지원자: 자동 완성 검색어 5개를 고르는 기준은 무엇입니까?

👩‍🎓 지원자: 맞춤법 검사 기능도 제공해야 합니까?

👩‍🎓 지원자: 질의는 영어 입니까?

👩‍🎓 지원자: 대문자나 특수 문자도 처리해야 하나요?

👩‍🎓 지원자: 얼마나 많은 사용자를 지원해야 합니까?

2. 문제 이해 및 설계 범위 확정

- 검색어 자동 완성 시스템 설계 시 고려해야 할 요구사항

1. 빠른 응답 속도
   - 사용자가 검색어 입력에 따라 자동 완성 검색어도 빨리 표기되어야 함
2. 연관성
   - 자동 완성되어 출력되는 검색어는 사용자가 입력한 단어와 연관성이 있어야 함
3. 정렬
   - 시스템의 계산 결과 인기도 등의 순위 모델에 의해 정렬되어야 함
4. 규모 확장성
   - 시스템은 많은 트래픽을 감당할 수 있도록 확장이 가능해야 함
5. 고가용성
   - 시스템의 일부에 장애가 발생하거나, 느려져도 시스템은 계속 사용이 가능해야 함

2. 문제 이해 및 설계 범위 확정

- 면접관의 대화를 통한 개략적인 규모 추정

1. 하루 이용자 수
   - 하루 이용자 수는 천만명, 한 명의 사용자는 매일 10건의 검색을 수행

2. 질의할 때마다, 20바이트의 데이터를 입력한다고 가정함
   - 문자 인코딩 방법으로 ASCII 코드를 사용한다고 하면, 1문자는 1바이트로 측정됨
   - 평균적으로 5글자로 이루어지며 4개의 단어로 이루어진다면 한번의 질의당 20바이트로 계산

3. 신규 데이터 추가
   - 질의 가운데 20% 정도가 신규 데이터라고 가정함
   - 약 0.4GB의 신규 데이터가 시스템에 추가됨

3. 개략적 설계안 제시 및 동의 구하기

- 검색어 자동 완성 시스템은 크게 **데이터 수집**과 **질의 서비스**로 2가지로 구분할 수 있음

1. 데이터 수집

- 사용자가 입력한 질의를 실시간으로 수집하는 서비스

-> 해당 부분은 보다 상세 설계에서 살펴봄

2. 질의 서비스

- 입력된 질의에 대해서 5개의 인기 검색어를 정렬해서 제공하는 서비스

3. 개략적 설계안 제시 및 동의 구하기

1. 데이터 수집

- 질의문과 사용빈도를 저장하는 빈도 테이블을 생성

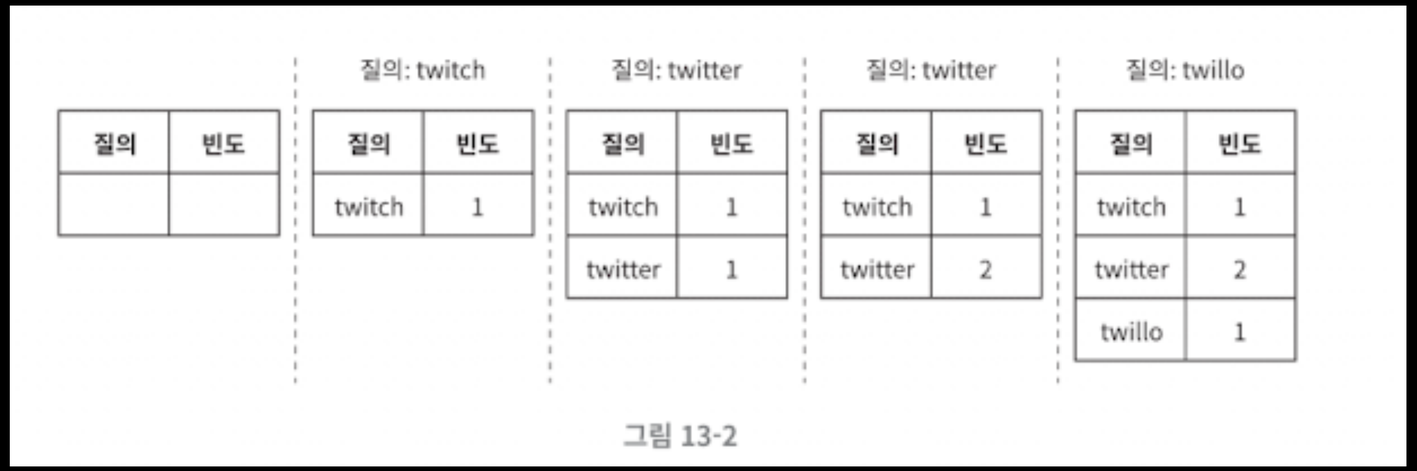


그림 13-2

3. 개략적 설계안 제시 및 동의 구하기

2. 질의 서비스

- 빈도 테이블에 있는 필드를 사용하여 사용자가 입력한 질의문의 빈도를 확인가능함

- **Query 필드 = 질의문을 저장하는 필드**
- **Frequency 필드 = 질의문이 사용된 빈도를 저장하는 필드**

query	freuqency
twitter	35
twitch	29
twilight	25
twin peak	21
twitch prime	18
twitter search	14
twillo	10
twin peak sf	8

표 13-1

3. 개략적 설계안 제시 및 동의 구하기

2. 질의 서비스

- 사용자가 글자 입력시, 다음과 같이 Top5 자동완성 검색어가 표시되어야 함

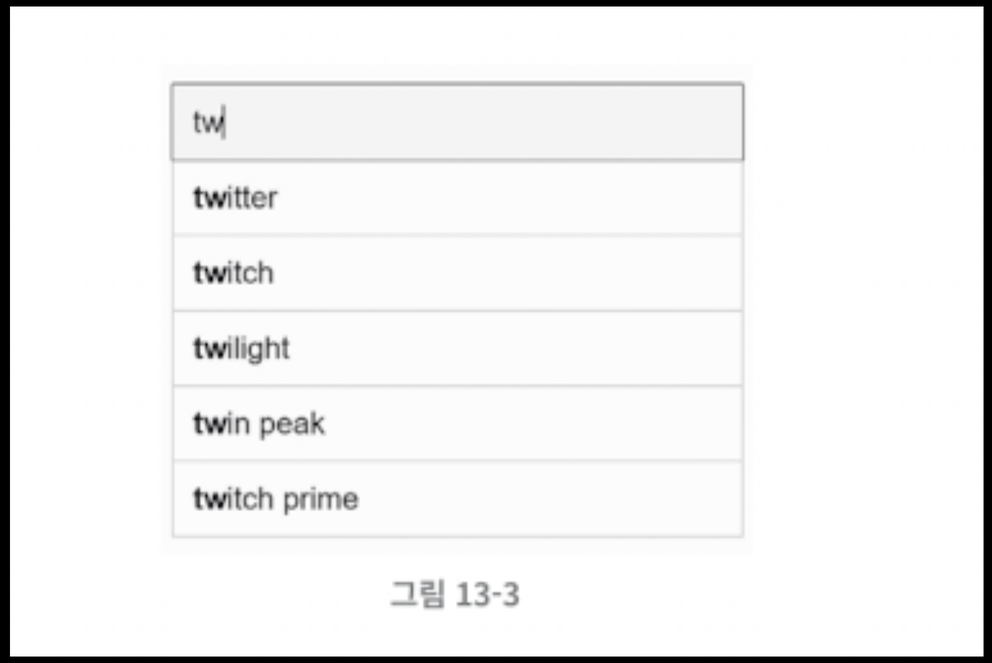


그림 13-3

- SQL 구문

```
SELECT * FROM frequency_table
WHERE query LIKE `prefix%`
ORDER BY frequency DESC
LIMIT 5
```

4. 상세 설계

- 대략적인 설계안을 통해 아래의 컴포넌트들을 보다 살펴보고, 상세히 설계함

1. 트라이(trie) 자료구조

2. 데이터 수집 자료구조

3. 질의 서비스

4. 규모 확장이 가능한 저장소

5. 트라이 연산

4. 상세 설계

- 대략적인 설계안을 통해 아래의 컴포넌트들을 보다 살펴보고, 상세히 설계함

1. 트라이(trie) 자료구조

2. 데이터 수집 자료구조

3. 질의 서비스

4. 트라이 연산

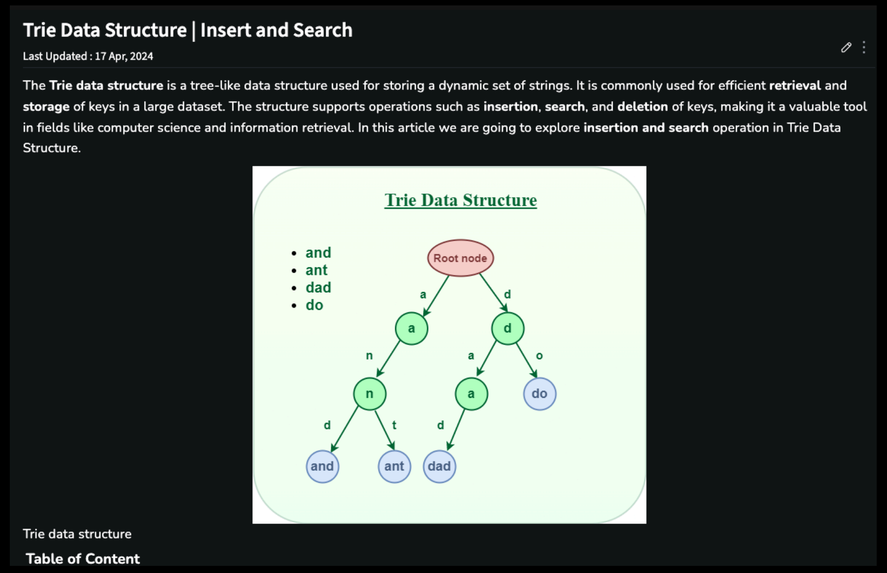
5. 규모 확장이 가능한 저장소

4. 상세 설계

1. 트라이(trie) 자료구조

대략적인 설계안에서는 관계형 DB를 저장소로 사용했지만, 가장 인기 있는 5개의 질의문을 골라내는 것에 효율적이지 않음

-> “트라이”라는 자료구조를 사용함



Trie | (Insert and Search) - GeeksforGeeks

트라이 자료구조의 특징

1. 트리형태의 자료구조
2. 해당 트리의 루트 노드는 빈 문자열을 나타냄
3. 각 노드는 글자 하나를 저장하며, 26개의 자식 노드를 가질 수 있음
4. 각 트리 노드는 하나의 단어 혹은 접두어 문자열을 나타냄

4. 상세 설계

1. 트라이(trie) 자료구조

예시

🔎 기존의 빈도 테이블

query	frequency
tree	10
try	29
true	35
toy	14
wish	25
win	50

표 13-2

👉 트라이 노드 사용

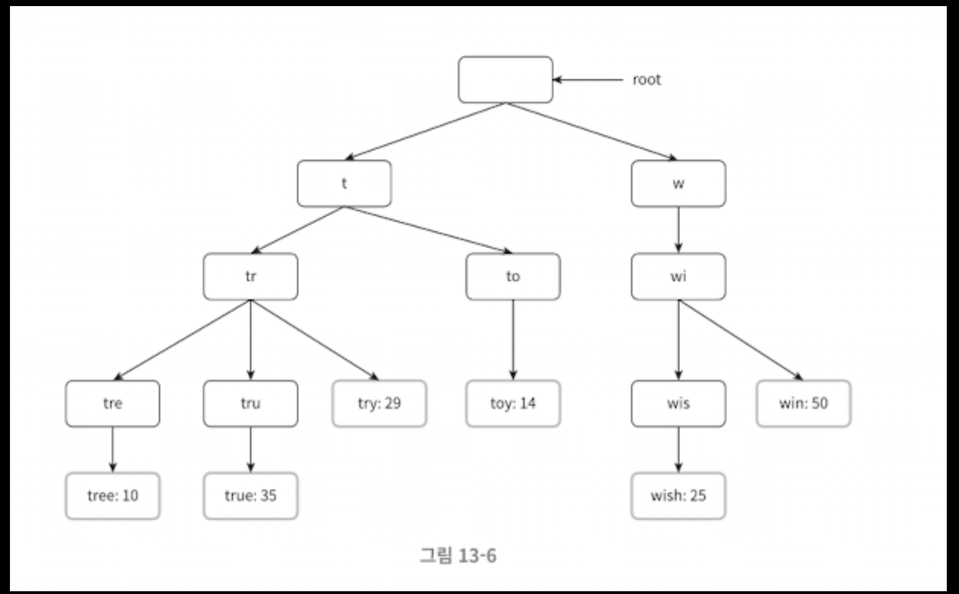


그림 13-6

4. 상세 설계

1. 트라이(trie) 자료구조를 이용한 자동 완성 구현

P: 접두어의 길이

N: 트라이 안에 있는 노드의 갯수

C: 주어진 노드의 자식 갯수

예시 - 가장 많이 사용된 질의어 K개의 갯수 검색

1. 해당 접두어를 표현하는 노드를 찾음, 시간 복잡도 = O(p)
2. 해당 노드부터 시작하는 하위 트리를 탐색하여 모든 유효 노드를 찾음, 시간 복잡도 = O(c)
3. 해당 유효 노드들을 정렬하여 가장 인기 있는 검색어 K개를 찾음, 시간 복잡도 = O(clogc)

4. 상세 설계

1. 트라이(trie) 자료구조를 이용한 자동 완성 구현

예시 = 가장 인기 있는 질의어 2개, 검색창에 “BE”를 검색했을 때

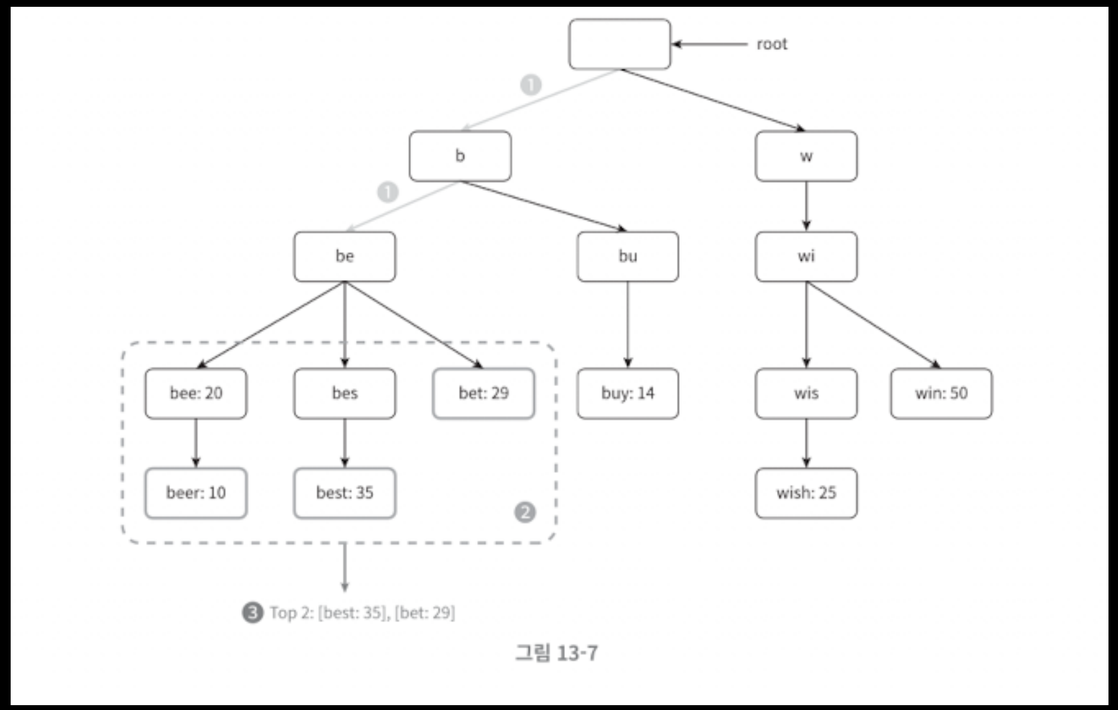


그림 13-7

총 시간 복잡도 = O(p) + O(c) + O(clogc)

유효 노드를 정렬하여 가장 사용빈도가 높은 **best, bet**를 선택하여 반환

4. 상세 설계

1. 트라이(trie) 자료구조를 이용한 자동 완성 구현의 주의점

최악의 경우, K개의 결과를 얻으려고 할 때, 전체 트라이를 모두 검색하는 문제점이 발생함

해결책

1. 접두어의 최대 길이를 제안

- 접두어 노드를 찾는 단계의 시간 복잡도 개선

-> 시간 복잡도 = O(1)로 개선

2. 노드에 인기 검색어 캐시

- 각 노드에 k개의 인기 검색어를 저장해두면 전체 트라이를 검색하는 일을 방지

4. 상세 설계

2. 노드에 인기 검색어 캐시

- 각 노드에 질의어를 저장할 공간이 많이 필요한다는 단점도 있지만, 빠른 응답 속도라는 장점이 있음

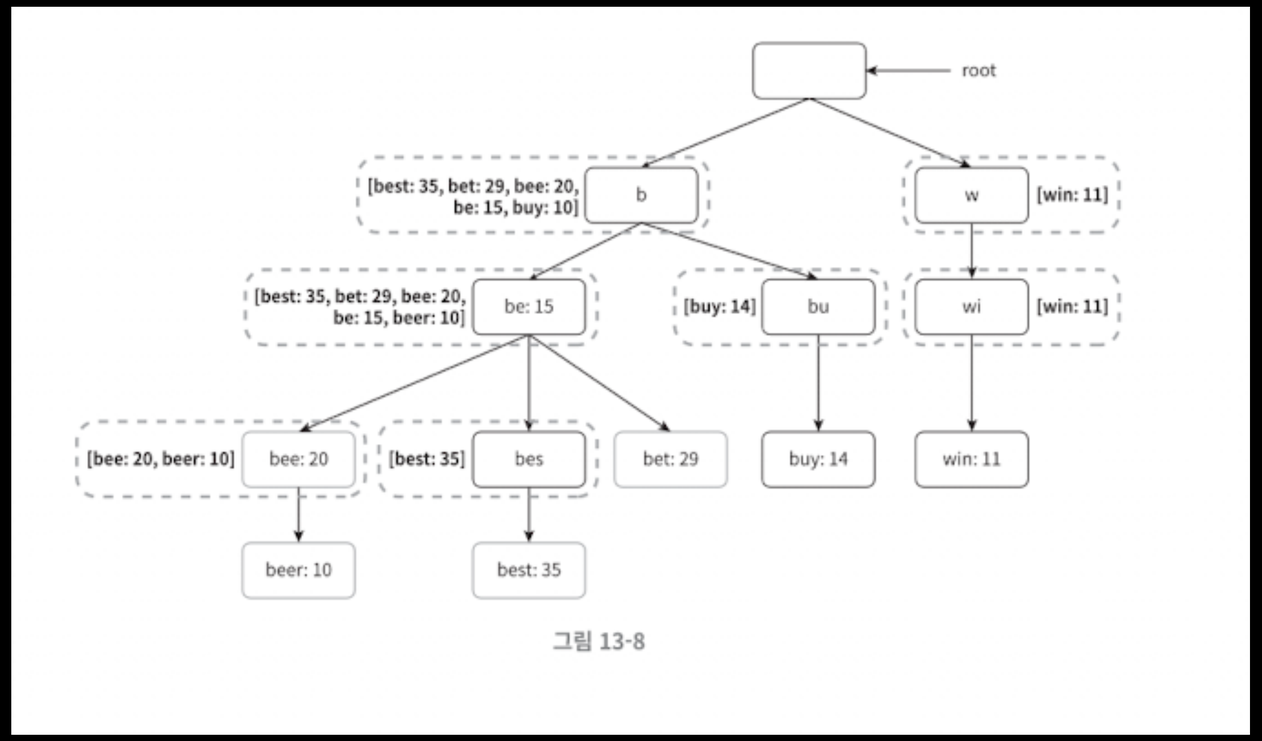


그림 13-8

개선 결과

- 접두어 노드를 찾는 시간 복잡도는 O(1)로 개선됨
- 최고 인기 검색어 5개를 질의하는 시간 복잡도도 O(1)로 개선됨

-> 전체 알고리즘의 시간복잡도는 O(1)로 개선됨

4. 상세 설계

2. 노드에 인기 검색어 캐시

- 각 노드에 질의어를 저장할 공간이 많이 필요한다는 단점도 있지만, 빠른 응답 속도라는 장점이 있음

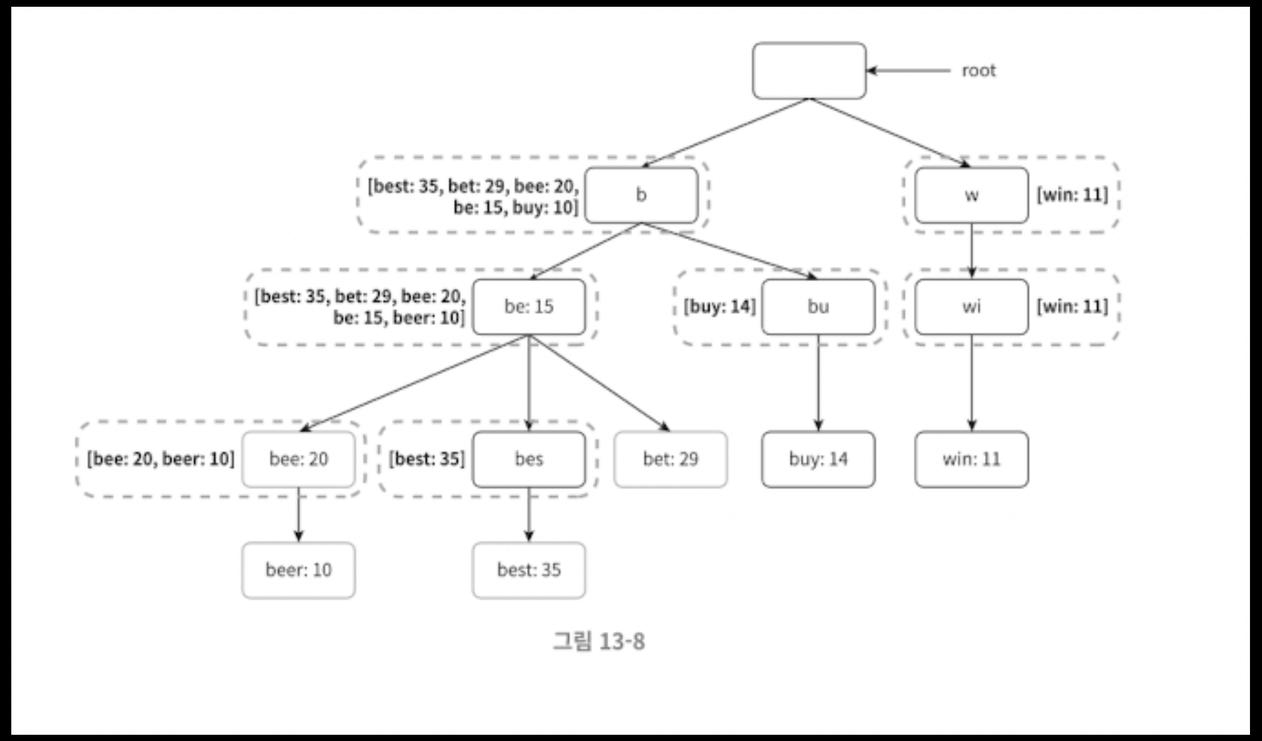


그림 13-8

개선 결과

- 접두어 노드를 찾는 시간 복잡도는 O(1)로 개선됨
- 최고 인기 검색어 5개를 질의하는 시간 복잡도도 O(1)로 개선됨

-> 전체 알고리즘의 시간복잡도는 O(1)로 개선됨

4. 상세 설계

2. 노드에 인기 검색어 캐시

- 각 노드에 질의어를 저장할 공간이 많이 필요한다는 단점도 있지만, 빠른 응답 속도라는 장점이 있음

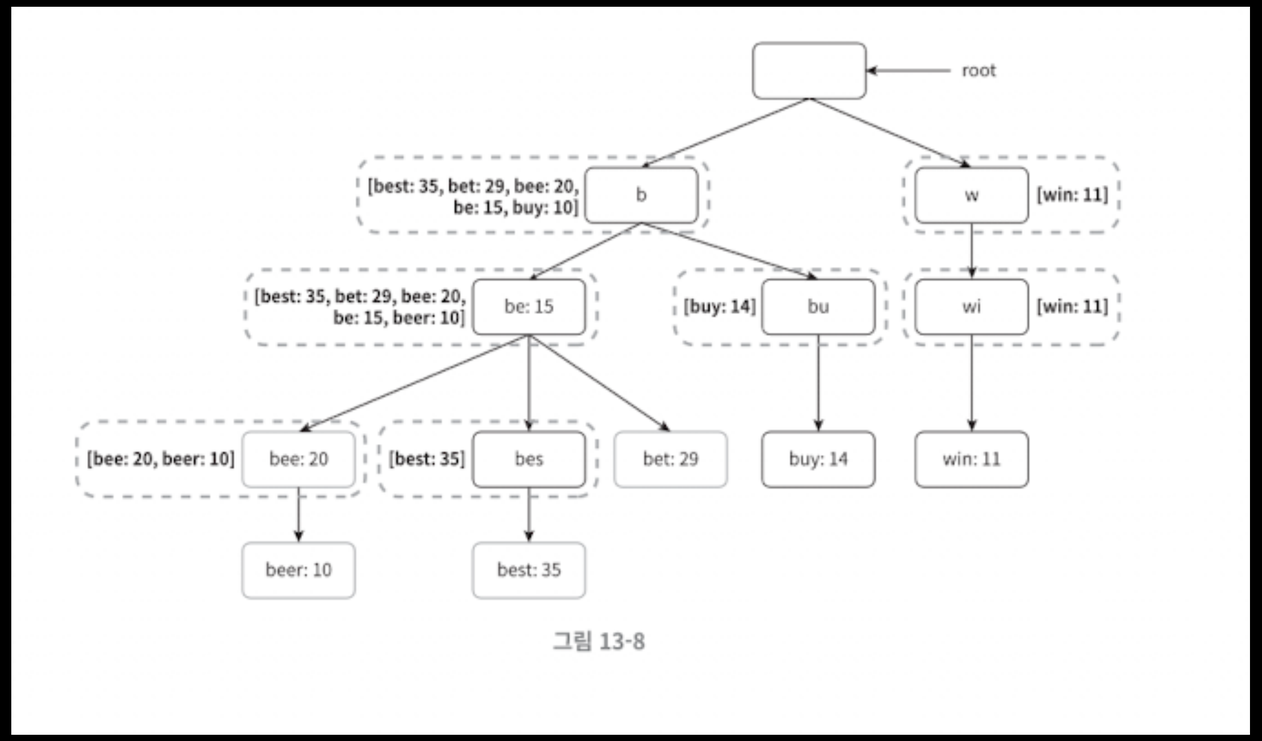


그림 13-8

개선 결과

- 접두어 노드를 찾는 시간 복잡도는 O(1)로 개선됨
- 최고 인기 검색어 5개를 질의하는 시간 복잡도도 O(1)로 개선됨

-> 전체 알고리즘의 시간복잡도는 O(1)로 개선됨

4. 상세 설계

2. 데이터 수집 자료구조

- 사용자가 검색창에 타이핑할 때마다 실시간으로 데이터를 수정할 때의 단점

1. 매일 수천만 건의 질의가 입력될 것인데, 그때마다 트라이를 갱신하면 질의 서비스는 느려짐
2. 트라이가 만들어지면, 인기 검색어는 자주 변동되지 않아, 트라이를 자주 갱신할 필요가 없음

-> 트라이를 만드는데 사용되는 데이터는 보통 데이터 분석 서비스나 로깅 서비스를 이용하여 설계함

4. 상세 설계

2. 데이터 수집 자료구조

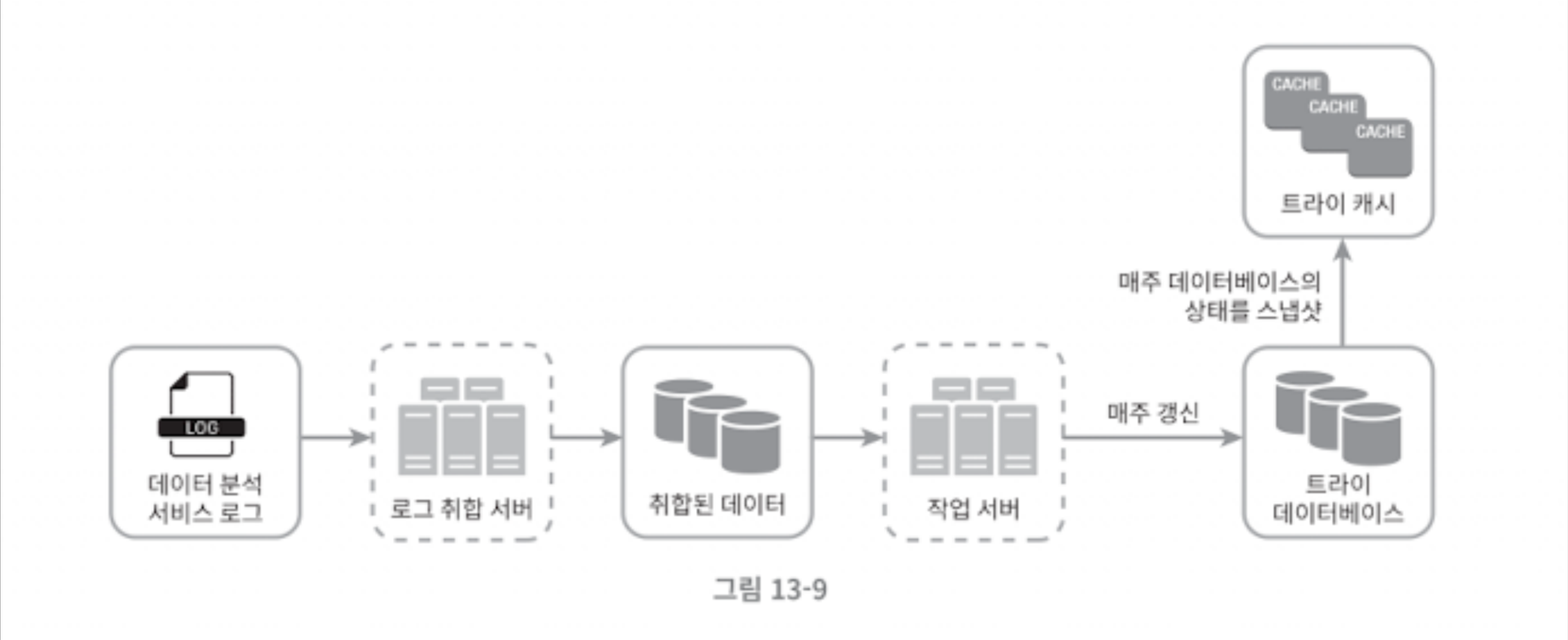


그림 13-9

4. 상세 설계

2. 데이터 수집 자료구조

데이터 분석 서비스 로그

query	time
tree	2019-10-01 22:01:01
try	2019-10-01 22:01:05
tree	2019-10-01 22:01:30
toy	2019-10-01 22:02:22
tree	2019-10-02 22:02:42
try	2019-10-03 22:03:03

표 13-3

- 검색창에 입력된 질의에 대한 데이터 원본이 저장되는 장소
- 새로운 데이터가 추가될 뿐, 수정은 이루어지지 않으며 로그 데이터에는 인덱스를 걸지 않고 사용함

4. 상세 설계

2. 데이터 수집 자료구조

로그 취합 서버

- 수집되는 데이터의 형식은 제각각인 경우가 많아 해당 데이터를 취합하여 가공할 수 있게 준비

취합된 데이터

query	time	frequency
tree	2019-10-01	12000
tree	2019-10-08	15000
tree	2019-10-15	9000
toy	2019-10-01	8500
toy	2019-10-08	6256
toy	2019-10-15	8866

표 13-4

- 매주 취합된 데이터를 의미함
- time필드 = 해당 주가 시작한 날짜
- frequency필드 = 해당 질의가 해당 주에 사용된 횟수의 합

4. 상세 설계

2. 데이터 수집 자료구조

작업 서버

- 트라이 자료구조를 만들고, 트라이 데이터베이스에 저장하는 역할을 담당함

트라이 캐시

- 분산 캐시 시스템으로 트라이 데이터를 메모리에 유지하여 읽기 연산 성능 향상 시킴
- 매주 트라이 데이터베이스의 스냅샷을 떠서 갱신함

4. 상세 설계

2. 데이터 수집 자료구조

트라이 DB

1. 문서 저장소를 사용하는 방법

- 매주 새 트라이를 만듬으로써, 주기적으로 트라이를 직렬화하여 DB에 저장할 수 있음
- 몽고 DB와 같은 문서 저장소를 활용하면 데이터를 편하게 저장할 수 있음

2. 키-값 저장소를 이용하는 방법

- 트라이에 보관된 모든 접두어를 해시 테이블 키로 변환
- 각 트라이 노드에 보관된 모든 데이터를 해시 테이블 값으로 변환

4. 상세 설계

2. 데이터 수집 자료구조

트라이 DB 2. 키-값 저장소를 이용하는 방법

- 각 트라이 노드는 하나의 <키,값> 쌍으로 변환됨

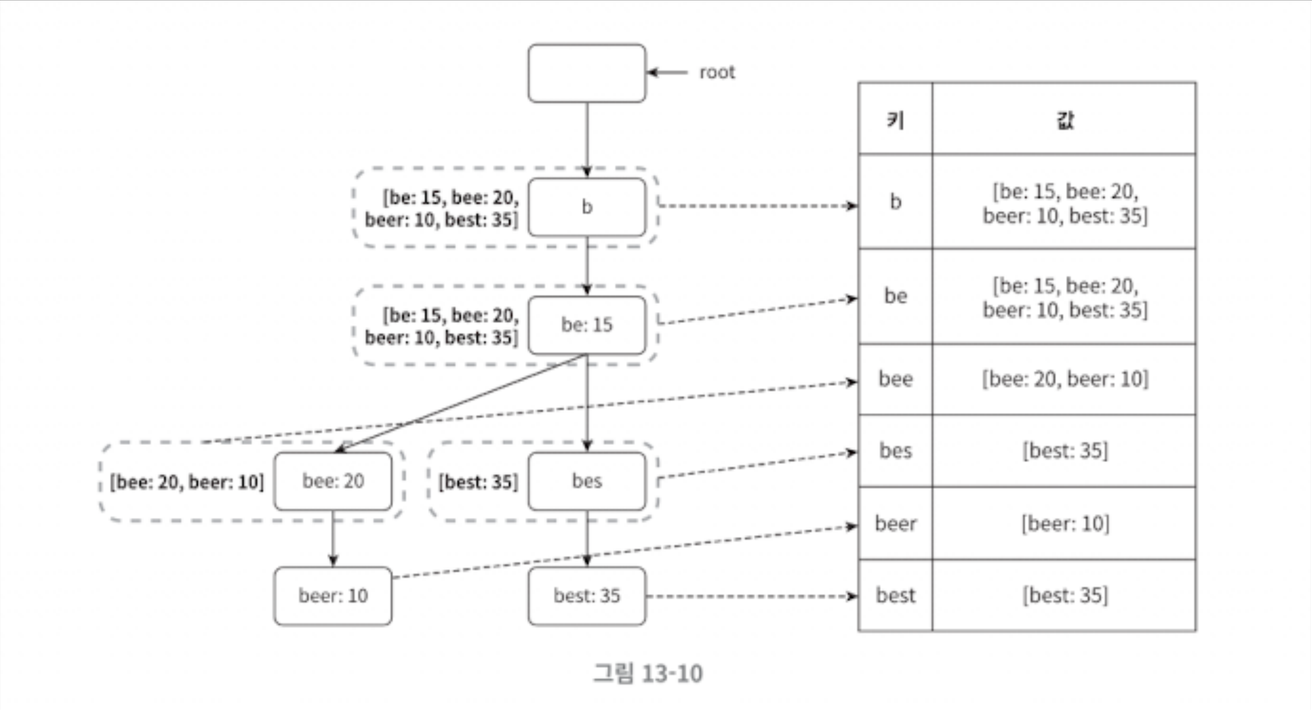


그림 13-10

4. 상세 설계

3. 질의 서비스

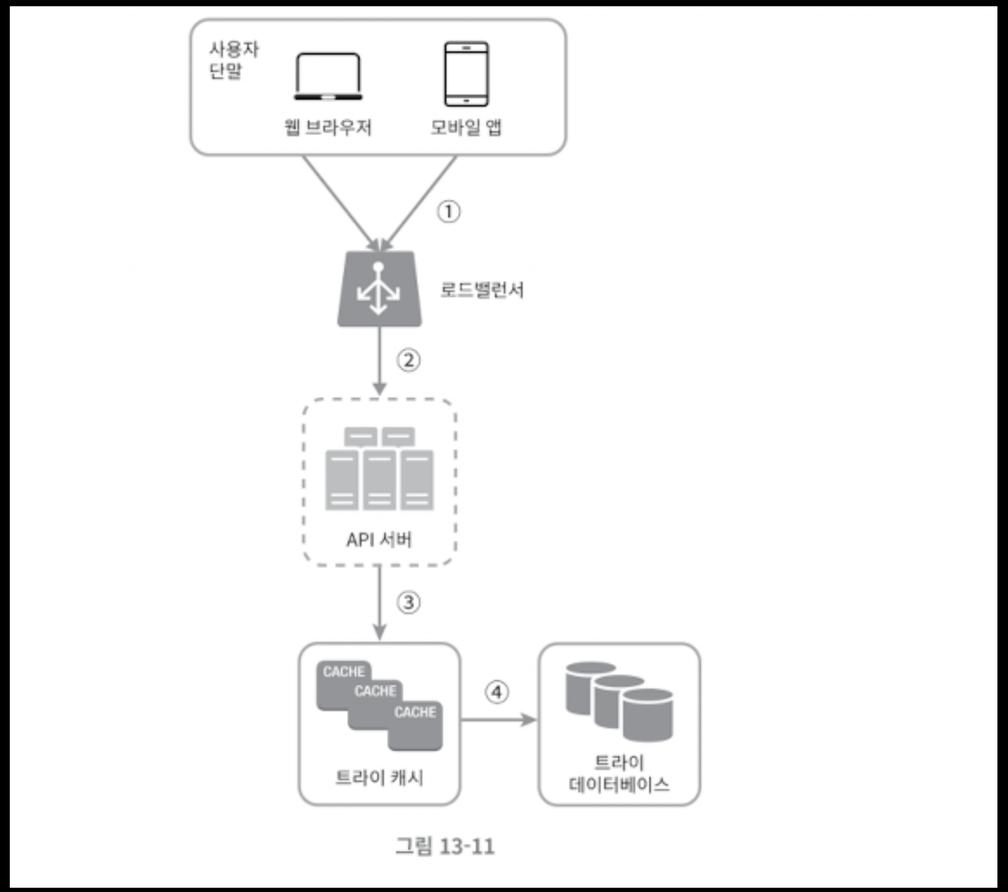


그림 13-11

1. 검색 질의가 로드밸런서로 전송됨

2. 로드 밸런서는 해당 질의를 API 서버로 보냄

3.API 서버는 트라이 캐시에서 데이터를 가져와, 해당 요청에 대한 자동완성 검색어 제안응답을 구성함

4.데이터가 트라이 캐시에 없는 경우, 데이터를 DB와 캐시에서 가져와 채움

4. 상세 설계

4. 트라이 연산

- 트라이 연산은 트라이 생성과 트라이 갱신으로 구분됨

트라이 생성

- 트라이 생성은 작업서버가 담당하며, 데이터 분석 서비스의 로그나 데이터베이스로부터 취합된 데이터를 이용함

트라이 갱신

1. 매주 한번 트라이를 갱신하는 방법
2. 트라이의 각 노드를 개별적으로 갱신하는 방법

-> 트라이가 작을 때는 고려할 만하지만, 성능이 좋지 못하다는 단점이 있음

4. 상세 설계

4. 트라이 연산 - 트라이 갱신

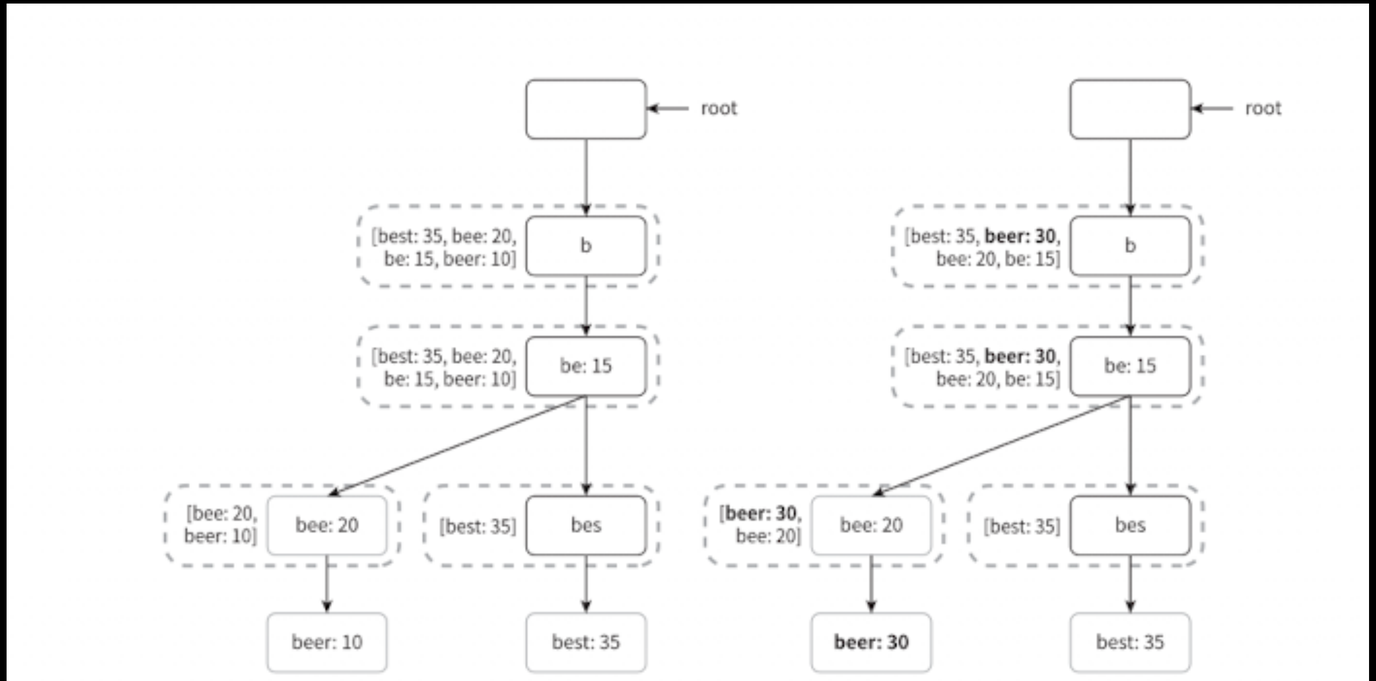


그림 13-13

4. 상세 설계

4. 트라이 연산 - 검색어 삭제

- 폭력적인 단어, 민감한 단어의 경우 자동완성에서 제외해야 함
- 트라이 캐시 앞에 필터 계층을 두고, 필터 규칙에 따라 부적절한 질의어가 반환되지 않도록 설정함

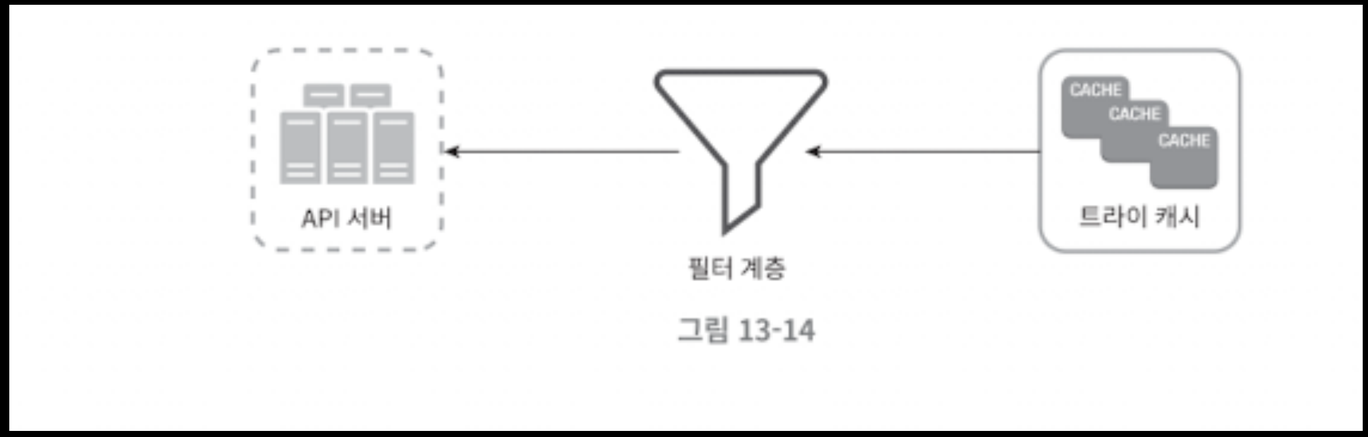


그림 13-14

4. 상세 설계

5. 저장소 규모 확장

- 트라이의 크기가 한 서버에 넣기에 너무 큰 경우에도 대응할 수 있도록 확장성 문제를 고려하는 경우를 의미함
- 영어만 지원하면 되기 때문에, 간단하게 첫글자를 기준으로 샤딩하는 방법을 고려함

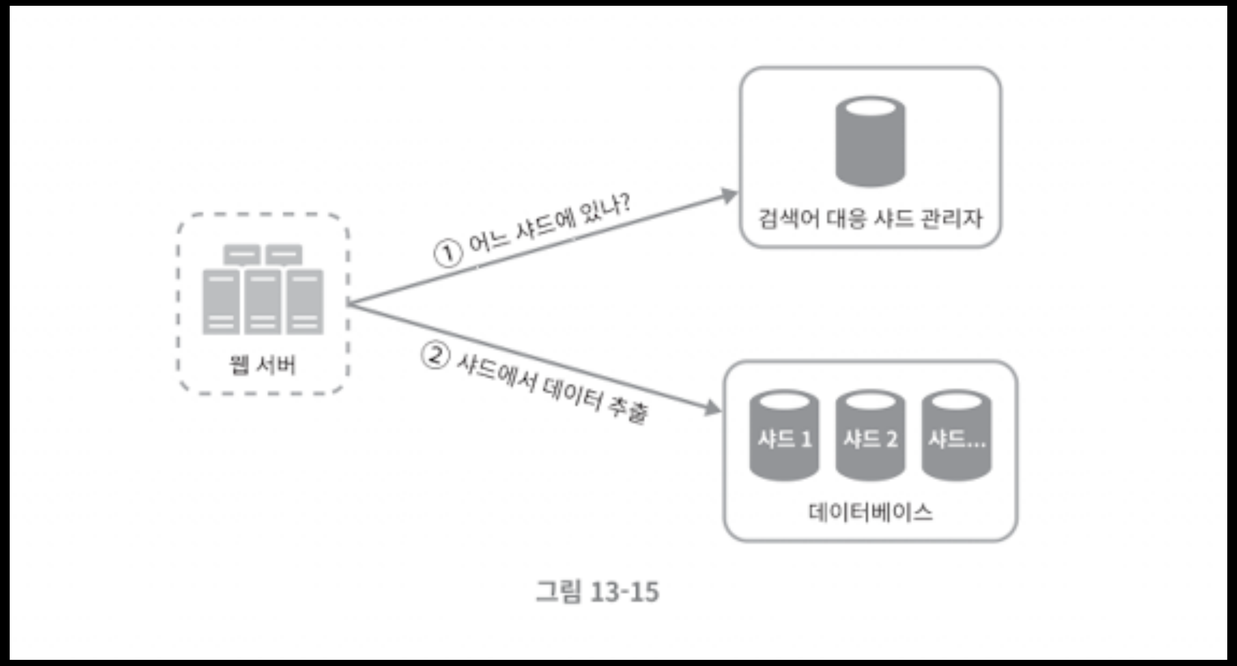


그림 13-15

5. 마무리

- 해당 상세 설계가 마치고 나면, 다음과 같은 사항들을 고려할 수 있음

1. 다국어 지원
   - 비영어권 국가에서 사용하는 언어를 지원하려면 유니코드로 데이터를 저장함
2. 국가별 인기 검색어 순위
   - 국가별로 다른 트라이를 사용
   - 트라이를 CDN에 저장하여 응답속도를 높이는 방법도 고려

5. 마무리

- 해당 상세 설계가 마치고 나면, 다음과 같은 사항들을 고려할 수 있음

3. 실시간으로 변하는 검색어의 추이를 반영

- 한번에 모든 데이터를 동시에 사용할 수 없을 가능성을 고려해야 하며, 지속적으로 데이터가 생성되어 보다 특별한 종류의 시스템이 필요함으로 하둡 맵 리듀스, 아파치 스파이크 스트리밍, 아파치 카프카를 고려