[13장] 검색어 자동완성 시스템

가상 면접 사례로 배우는 대규모 시스템 설계 기초

이민석 / unchaptered

https://inblog.ai/monthly-cs
https://github.com/monthly-cs/2024-03-system-design-interview-1

1단계 | 요구사항 분석 | 문제 이해 및 설계 범위 확정

요구사항 파악 및 정리

1. 자동완성은 검색어의 **첫 부분**으로 진행
2. 자동완성 검색어 표시는 **5개**까지
3. 검색어 인기 순위는 **질의빈도**를 사용
4. 맞춤법 검사나 자동수정 미지원
5. 쿼리 질의는 **기본은 영어**지만 다국어 지원도 고려
6. 영어 소문자를 제외한 문자(특수문자 포함)는 지원하지 않음
7. 일간 능동 사용자(DAU)가 천만명
8. 100ms 이내의 빠른 응답속도 지원
9. 자동완성 출력 검색어의 연관성이 높아야함
10. 인기도 등의 몇가지 **순위 모델을**을 지원해야함
11. 고확장성, 고가용성 지원해야함

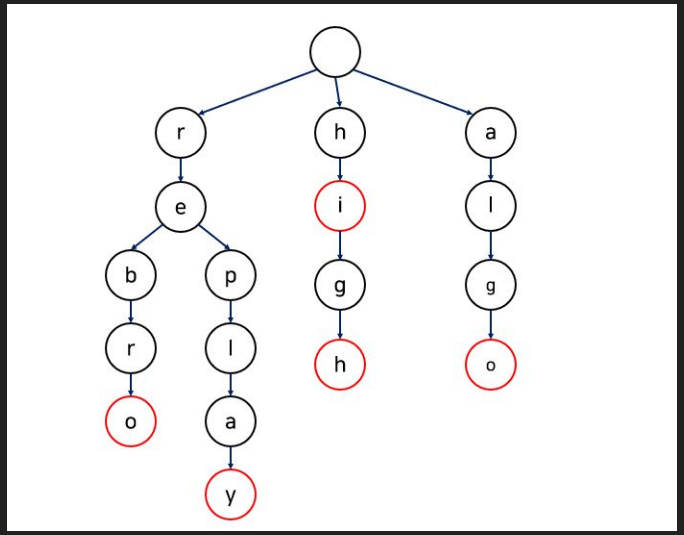
개략적 규모 추정

1. 일간 능동 사용자(DAU)는 10,000,000명(천만명)
2. 사용자 별 평균 일간 검색 횟수는 10회(십회)
3. 글자 입력 시 마다 백엔드에 자동 완성 요청 전송(1회당 20회 요청 전달)
4. 초당 QPS 추정치 = 10,000,000명 × 10회/일 × 20자 × 24시간 / 3600초 = **24,000회/초**
5. 최대 QPS 추정치 = QPS × 2 = **48,000회/초**
6. 신규 검색어 비율 = **20%**
7. 신규 저장 데이터량 = 10,000,000명 × 10회/일 × 20자 × 20% = **0.4GB**

개략적인 설계

1. 데이터 수집 서비스
    a. 사용자가 입력한 질의를 실시간으로 **수집**
2. 질의 서비스
    a. 주어진 질의에 다섯 개의 인기 검색어를 정렬해서 반환

트라이 자료 구조 - 1

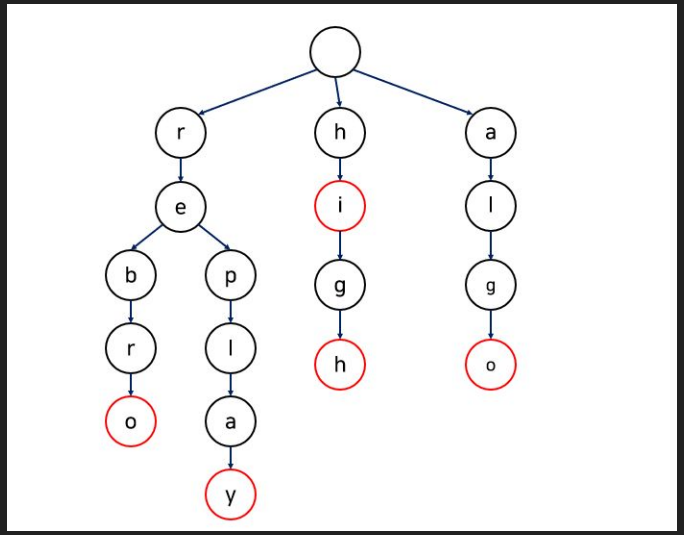


핵심 아이디어

- 트리 형태의 자료 구조
- 루트 노드는 빈 문자열
- 각 노드는 글자 하나를 저장하며 26개의 자식 노드를 가질 수 있음
- 각 트리 노드는 하나의 단어 또는 접두어 문자열을 표시 (이 때, 빈도 정보까지 저장)

https://velog.io/@kiloo/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%ED%8A%B8%EB%9D%BC%EC%9D%B4Trie-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0

트라이 자료 구조 - 2



사전 정의

- p : 접두어(prefix)의 길이
- n : 트라이 안에 있는 노드 갯수
- c : 주어진 노드의 자식 갯수

알고리즘

- 해당 접두어를 표현하는 노드 탐색 O(p)
- 유효 노드를 정렬하여 인기 검색어 k개 탐색 O(clogc)
- 접두어(p) 최대 길이를 제한하여, 노드에 인기 검색어 캐시 O(1)

https://velog.io/@kIloo/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%ED%8A%B8%EB%9D%BC%EC%9D%B4Trie-%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0

데이터 수집 서비스 - 1

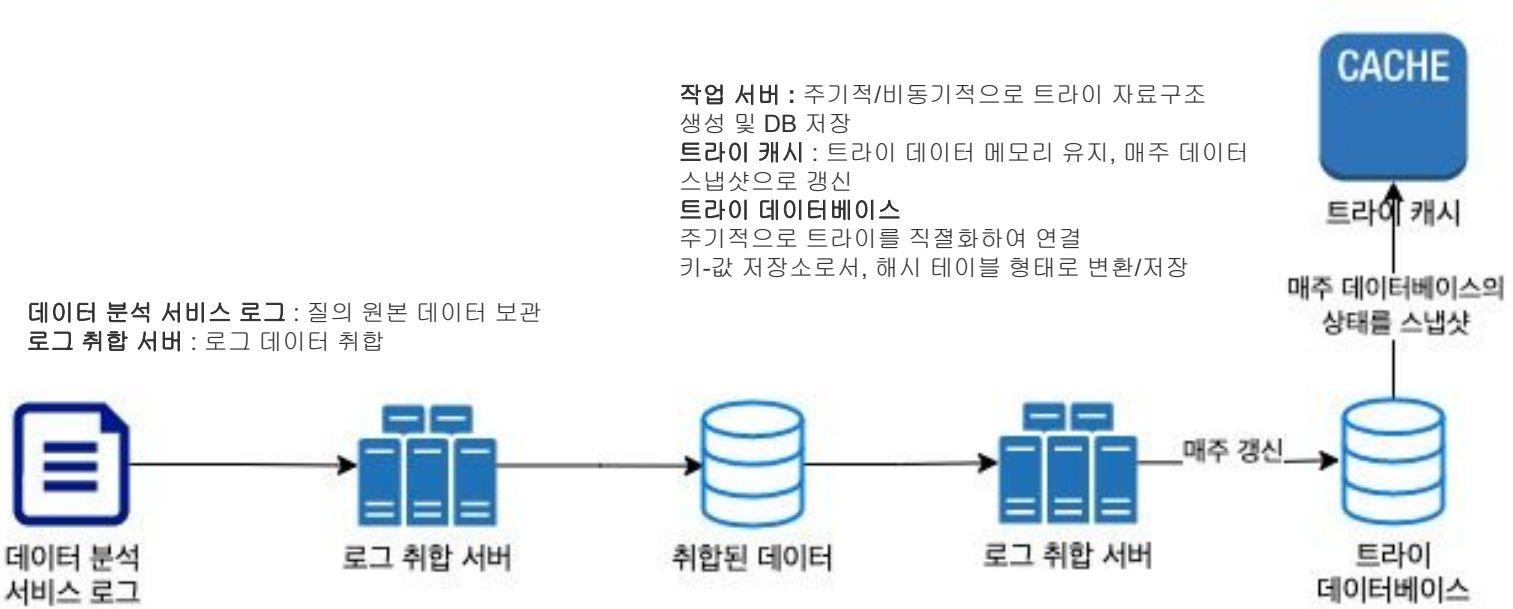
가설

- 매일 수천만 건의 질의 시 마다 트라이를 갱신 → **성능 저하 발생할 것**
- 트라이가 최초 생성 이후 → 인기 검색어가 자주 바뀌지 않을 수 있음 → **자주 갱신하지 않아도 될 것**

데이터 수집 서비스 - 2

작업 서버 : 주기적/비동기적으로 트라이 자료구조 생성 및 DB 저장
트라이 캐시 : 트라이 데이터 메모리 유지, 매주 데이터 스냅샷으로 갱신
트라이 데이터베이스
주기적으로 트라이를 직결화하여 연결
키-값 저장소로서, 해시 테이블 형태로 변환/저장

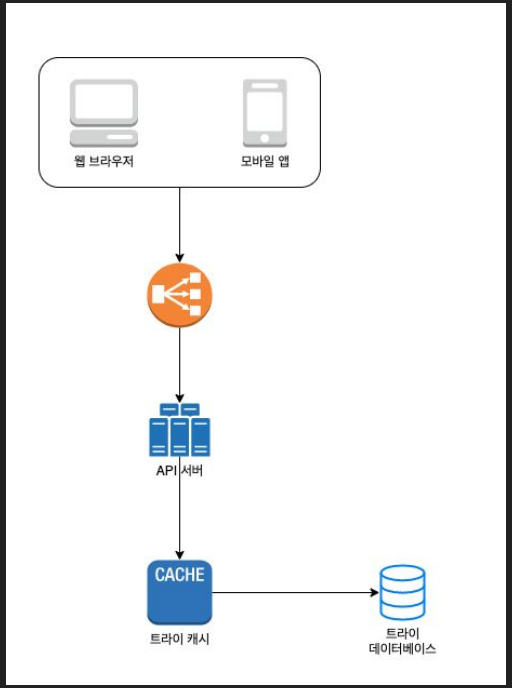
데이터 분석 서비스 로그 : 질의 원본 데이터 보관
로그 취합 서버 : 로그 데이터 취합



https://azderica.github.io/til/docs/dev/system-design-interview/ch13/

3단계 | 상세 설계

질의 서비스



최적화 방안

1. **AJAX 요청** : 페이지 새로고침 하지 않도록 변경
2. **브라우저 캐싱** : 브라우저 자체 캐싱 기능 활용
3. **데이터 샘플링** : N개의 요청 중에 1개만 로그를 남김

https://azderica.github.io/til/docs/dev/system-design-interview/ch13/

3단계 | 상세 설계

저장소 규모 확장

트라이가 커져서 문제가 된다면 규모확장성도 고민해야함

첫 글자를 기준으로 샤딩하는 방법을 사용하면

- 사용 가능한 서버가 26대(알파벳 종류 수)로 제한됨
- 또한 서버 별로 데이터가 균등하게 배포되지 않음

샤드 관리자가 과거의 질의 패턴을 분석하여 데이터를 샤딩하는 방법을 사용하면

- 위와 같은 문제를 겪지 않을 수 있음
- 단, **SPoF**는..?

마무리

- 다국어 지원 : 트라이에 유니코드 데이터 저장이 필요(한글 등)
- 국가별 인기 검색어 순위가 다른 경우 : 국가별 트라이 사용 필요 + CDN 캐싱
- 실시간으로 변하는 검색어 추이 반영을 위해
    - 샤딩을 통해 데이터 양 줄이기
    - 순위 모델을 변경해서 최근 검색어에 더 높은 가중치 부여
    - 데이터가 스트리밍 필요할 수도 있음
      (Haddop MapReduce, Apache Spark Streaming 등)

감사합니다.