가상 면접 사례로 배우는

# 대규모 시스템 설계 기초

이정민

# 13. 검색어 자동완성 시스템

# 1단계) 문제 이해 및 설계 범위 확정

■ 요구사항 파악

- 자동완성은 검색어의 첫 부분 or 중간 부분? 첫 부분
- 자동완성 검색어 표시 개수? 5개
- 자동완성 대상을 고르는 기준은? 검색어 인기 순위(질의 빈도)
- 맞춤법 검사나 자동수정 지원? 미지원
- 질의는 영어만? 영어. 다국어 지원도 고려 / 대문자 or 특수문자 처리 여부? 없음. 영어 소문자만
- 사용자 지원 수준? 일간 능동사용자 기준 천만명

# 1단계) 문제 이해 및 설계 범위 확정

■ 요구사항 정리

- 빠른 응답 속도 : 예) 페이스북의 경우 100밀리초 이내
- 연관성 : 자동완성 출력 검색어는 사용자가 입력한 단어와 연관된 것이어야 함
- 정렬 : 인기도 등의 순위 모델에 의해 정렬 되어야 함
- 규모 확장성 : 많은 트래픽을 감당할 수 있도록 확장 가능 해야 함
- 고가용성 : 장애, 성능 저하가 생겨도 시스템은 계속 사용 가능해야 함

# 1단계) 문제 이해 및 설계 범위 확정

■ 개략적 규모 추정

- 일간 능동 사용자 : 천만 명 / 사용자 별 평균 검색 건수 : 일 10건
- 질의 시 마다 평균 20바이트 데이터 입력을 가정함 (1문자 = 1바이트, 평균 4개 단어, 각 5개 글자)
- 검색창에 글자 입력 시 마다 백엔드에 자동완성 요청 전송 (1회 검색당 20건의 요청 전달)
- 대략 초당 24,000건 질의(QPS) 발생 (10,000,000 사용자 * 10질의/일 * 20자/24시간/3600초)
- 최대 QPS = QPS * 2 = 대략 48,000
- 질의 중 20%는 신규 검색어. 매일 0.4GB 신규 데이터가 시스템에 추가

# 2단계) 개략적 설계안 제시 및 동의 구하기

■ 주요 기능

- 데이터 수집 서비스 : 사용자가 입력한 질의를 실시간으로 수집
  - 질의/빈도 테이블
- 질의 서비스 : 주어진 질의에 다섯 개의 인기 검색어를 정렬하여 제공
  - 내림차순 정렬 5개 까지

# 3단계) 상세 설계

## ■ 트라이 자료구조

- 핵심 아이디어
  - 트리 형태의 자료구조. 루트 노드는 빈 문자열. 각 노드는 글자 하나를 저장하며 26개의 자식 노드를 가질 수 있음
  - 각 트리 노드는 하나의 단어 또는 접두어 문자열을 나타냄. 빈도 정보까지 저장
- 알고리즘
  - p: 접두어의 길이 / n : 트라이 안에 있는 노드 개수 / c : 주어진 노드의 자식 노드 개수 / k : 질의어 추천 개수
  - 해당 접두어를 표현하는 노드를 탐색. O(p) 해당 노드부터 하위 트리를 탐색하여 유효 노드 탐색. O(c)
  - 유효노드를 정렬하여 인기 검색어 k개를 탐색. O(clogc)
  - 최적화 방안 : 접두어(p) 최대 길이 제한. 노드에 인기 검색어 캐시 O(1)

# 3단계) 상세 설계

- 데이터 수집 서비스
  - 고려사항
    - 매일 수천만 건의 질입 시 마다 트라이를 갱신하면 성능 저하 발생
    - 일단 트라이가 만들어지고 나면 인기 검색어는 자주 바뀌지 않을 수 있음. 트라이는 자주 갱신 할 필요 없음
  - 개선 설계안 (p. 235 그림 13-9)
    - 데이터 분석 서비스 로그 : 질의 원본 데이터 보관 / 로그 취합 서버 : 로그 데이터 취합
    - 취합된 데이터 : query, time, frequency (주단위 사용된 횟수 합)
    - 작업 서버 : 주기적 비동기적 작업 실행. 트라이 자료구조 생성 및 데이터베이스 저장
    - 트라이 캐시 : 트라이 데이터 메모리 유지. 매주 데이터베이스의 스냅샷으로 갱신
    - 트라이 데이터베이스 : 문서저장소 - 주기적으로 트라이를 직력화하여 저장. 키-값 저장소 - 해시테이블 형태로 변환/저장

# 3단계) 상세 설계

## ■ 질의 서비스

- 개선 설계안 : p. 238 그림 13-11
  - 매일 수천만 건의 질입 시 마다 트라이를 갱신하면 성능 저하 발생
  - 일단 트라이가 만들어지고 나면 인기 검색어는 자주 바뀌지 않을 수 있음. 트라이는 자주 갱신 할 필요 없음
- 최적화 방안
  - AJAX 요청 : 자동 완성된 검색어 목록 조회 시 페이지 새로고침 할 필요 없음
  - 브라우저 캐싱
  - 데이터 샘플링 : N개 요청 중 1개만 로깅

# 3단계) 상세 설계

## ■ 트라이 연산

- 트라이 생성 : 작업 서버가 수행. 데이터 분석 서비스의 로그나 데이터베이스에서 취합된 데이터 사용
- 트라이 갱신
  - 매주 한번 갱신하는 방법 : 신규 트라이를 만든 다음에 기존 트라이를 대체
  - 트라이의 각 노드를 개별적으로 갱신하는 방법 : 성능이 좋지 않음
- 검색어 삭제 : 필터 계층 추가 (부적절한 질의 제거)

# 3단계) 상세 설계

## ■ 저장소 규모 확장

- 첫 글자를 기준으로 샤딩하는 방법
  - 사용 가능 서버 최대 26대로 제한
  - 26대 이상 사용하기 위해서는 샤딩을 계층적으로 적용 필요. 균등 배분하기가 불가능함
- 샤드 관리자
  - 과거 질의 데이터 패턴을 분석하여 샤딩
  - 검색어가 어느 저장소 서버에 저장되는 지 정보를 관리

# 4단계) 마무리

## ■ 추가 논의사항

- 다국어 지원 : 비영어권 국가 언어 지원을 위해 트라이에 유니코드로 데이터 저장 필요
- 국가별 인기 검색어 순위가 다른 경우 : 국가 별로 다른 트라이 사용. 트라이를 CDN에 저장
- 실시간으로 변하는 검색어 추이를 반영하려면
  - 트라이 구성에 많은 시간이 소요되어 매주 1회 트라이 갱신 하므로, 실시간 지원은 불가능함
  - 샤딩을 통하여 작업 대상 데이터 양을 줄여야 함. 순위 모델을 바꾸어 최근 검색어에 높은 가중치를 주어야 함
  - 데이터가 스트림 형태로 올 수 있다는 점 고려 필요. 스트림 프로세싱이 필요 할 수 있음