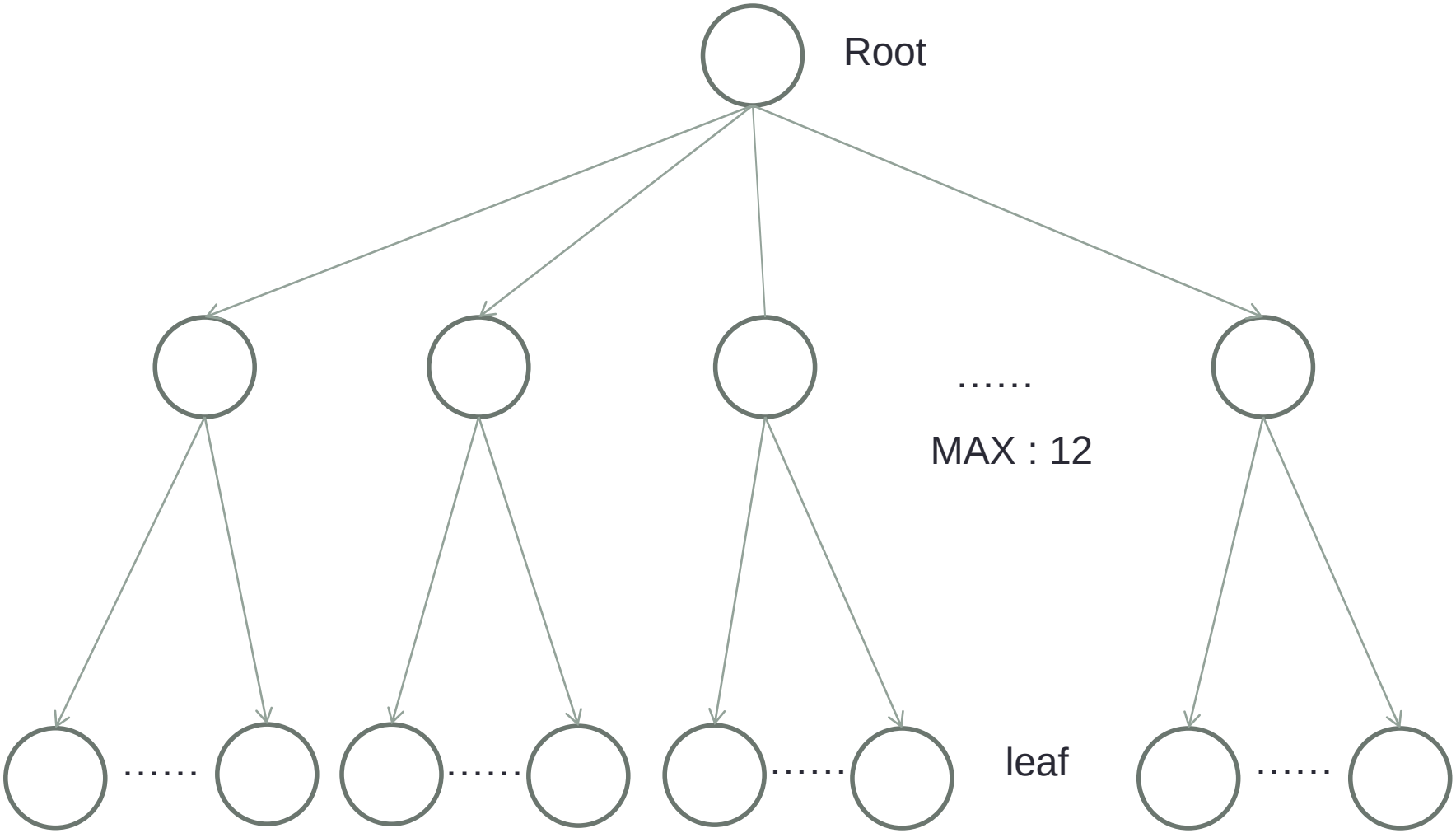
색인 처리 구조

1. 원천 데이터
   - 2010년 Wikipedia Dumb 데이터 56만건/2.6GB

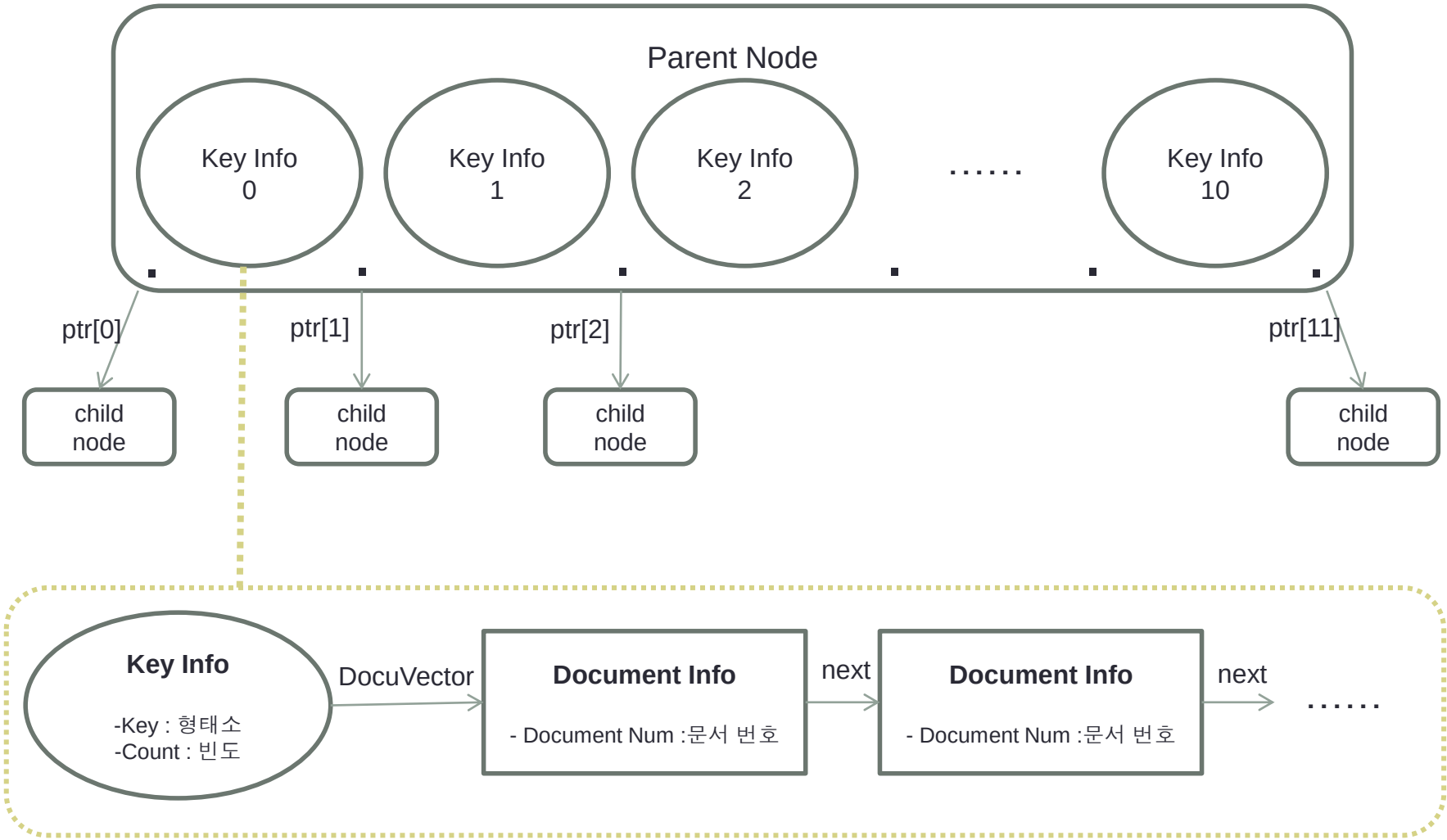
2. 역파일 대상 데이터 구조
   - (238789,0,'Bedrock_Records','House record label started by Nick Muir and John Digweed. It is based around a club night also called Bedrock. Basically, the artists use this label to expose the sounds that they like to the world. ','Bedrock Record Label and Club Night',0,'61.9.128.xxx','20010726075539',0)

3. 역파일 절차
   1) Index 추출
   2) 공백 기준 형태소 추출
      - 특수문자 제거
      - 대문자->소문자
      - html 태그 제거
   3) '형태소-색인 리스트' 구조로 리스트 생성
   4) B-Tree에 색인 정보 저장

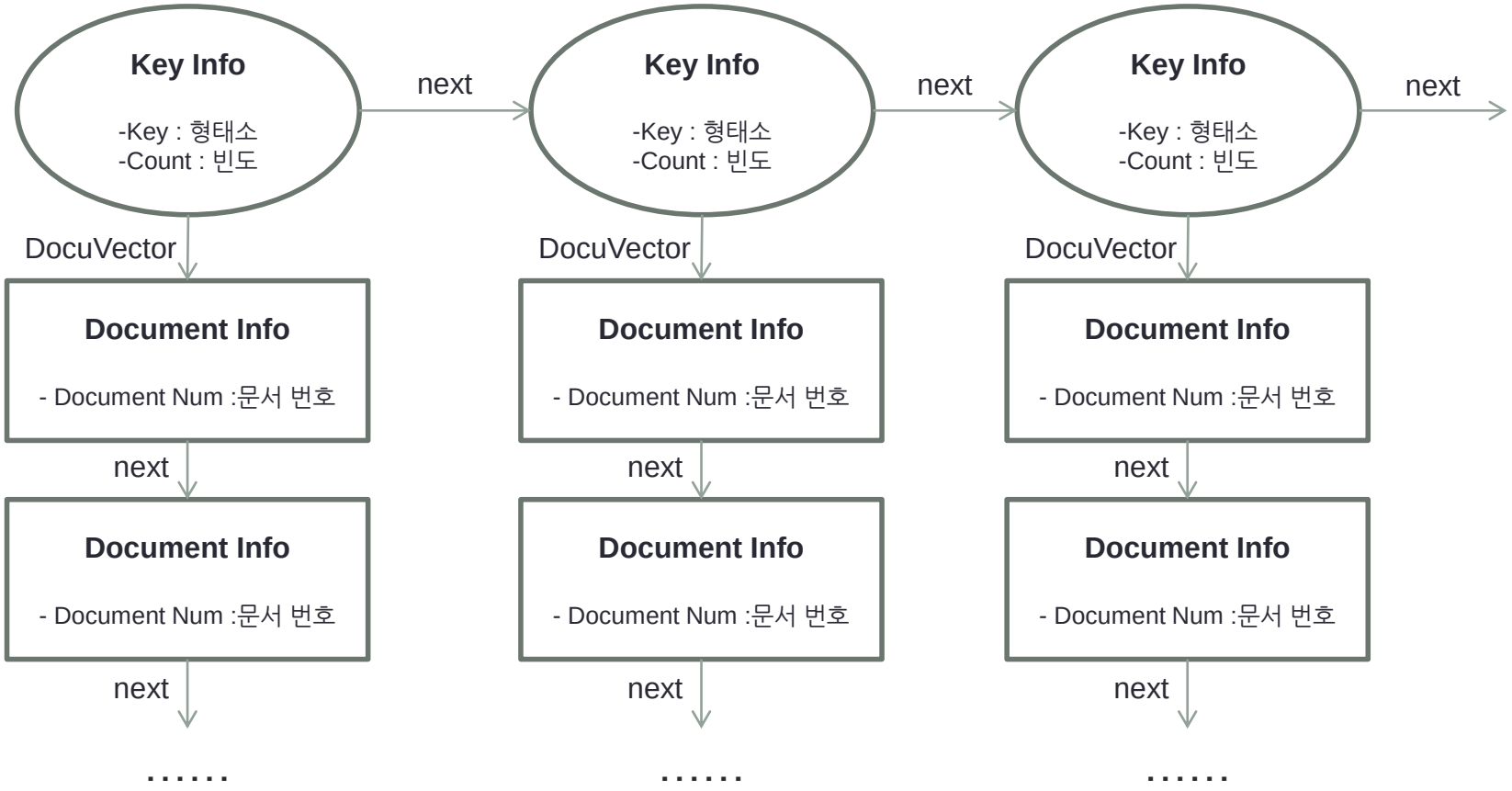
B-Tree 구조



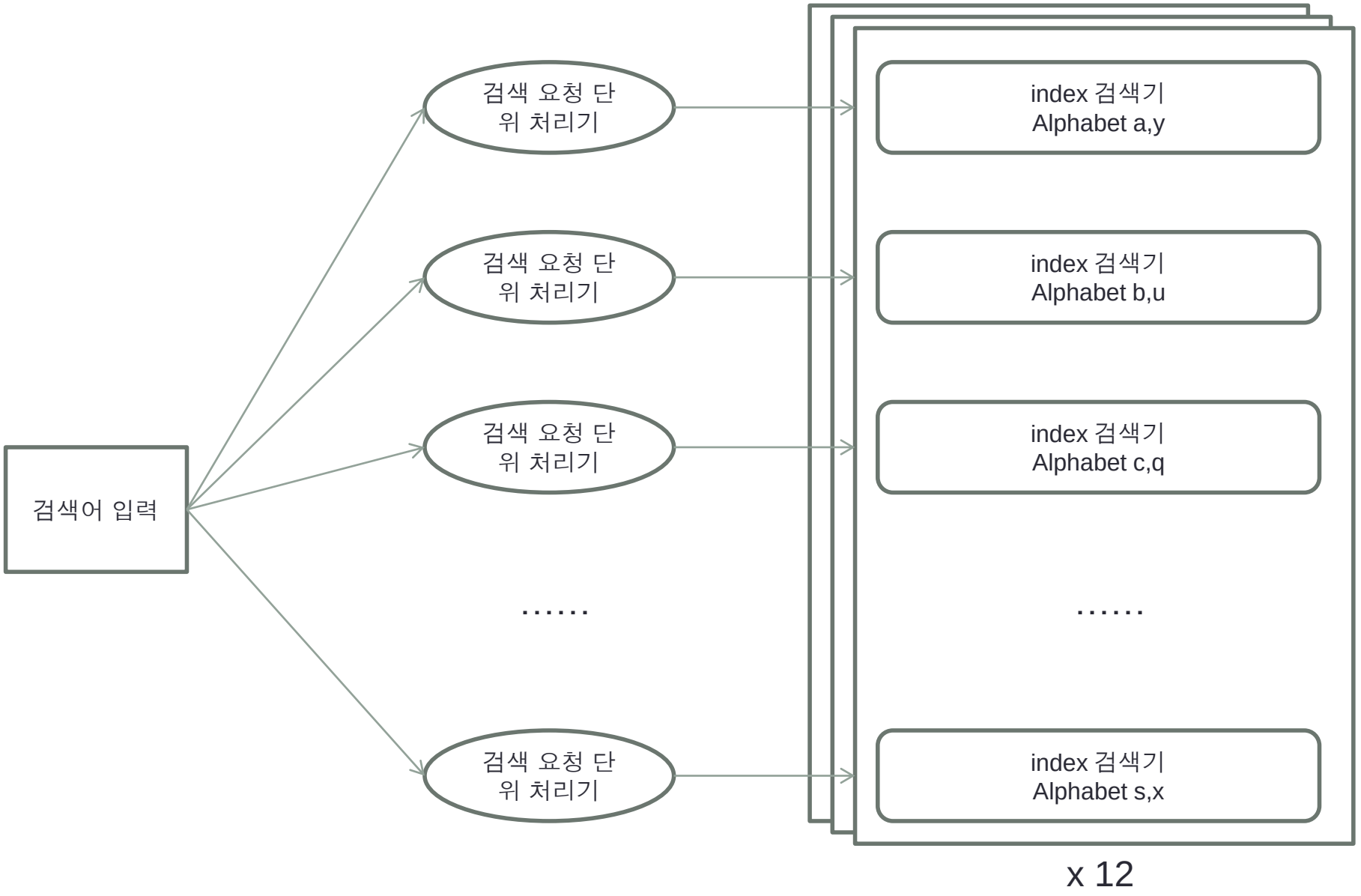
B-Tree node 데이터 구조



문서 역파일 및 검색어 입력 시 Data 구조



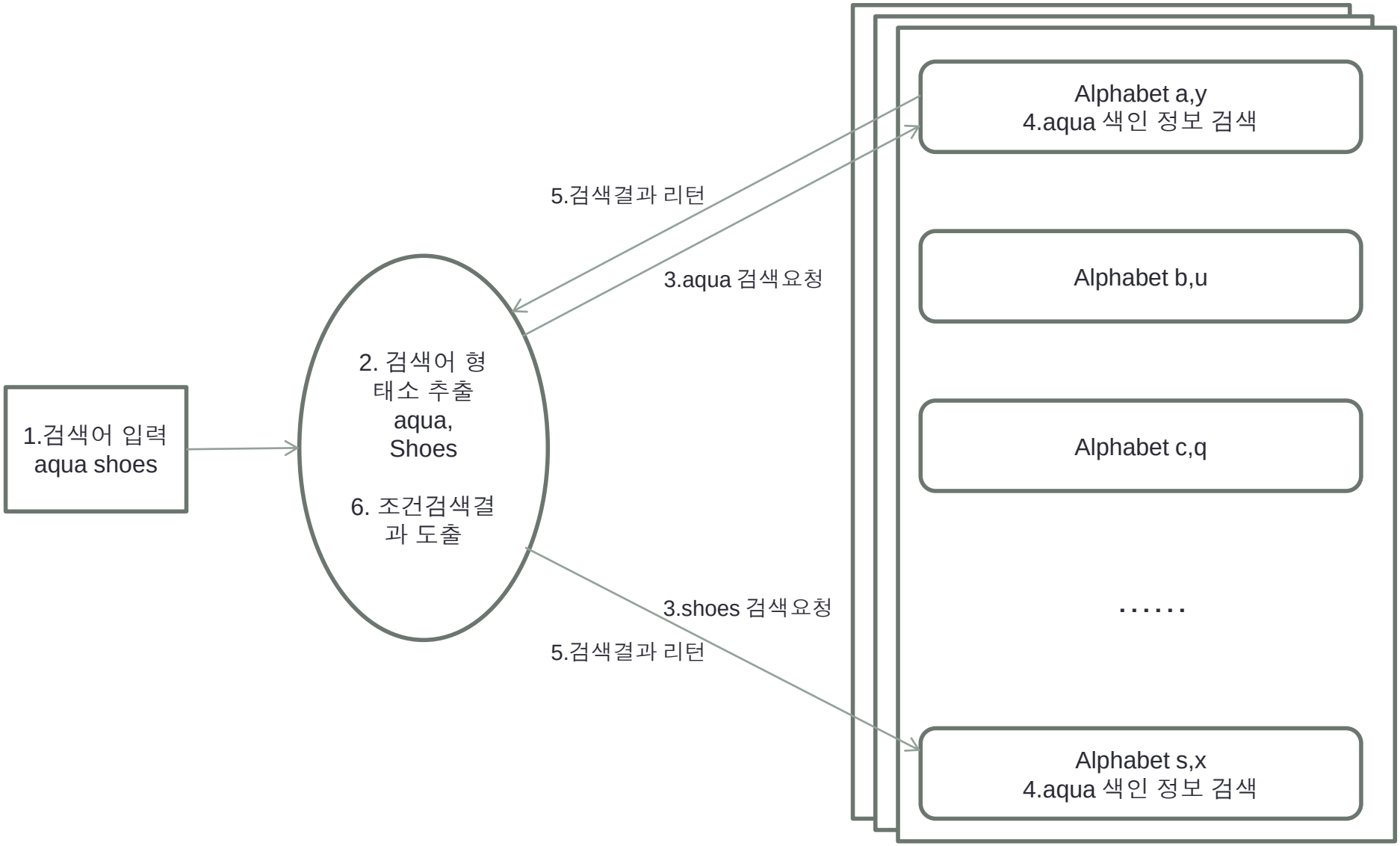
검색기 구조



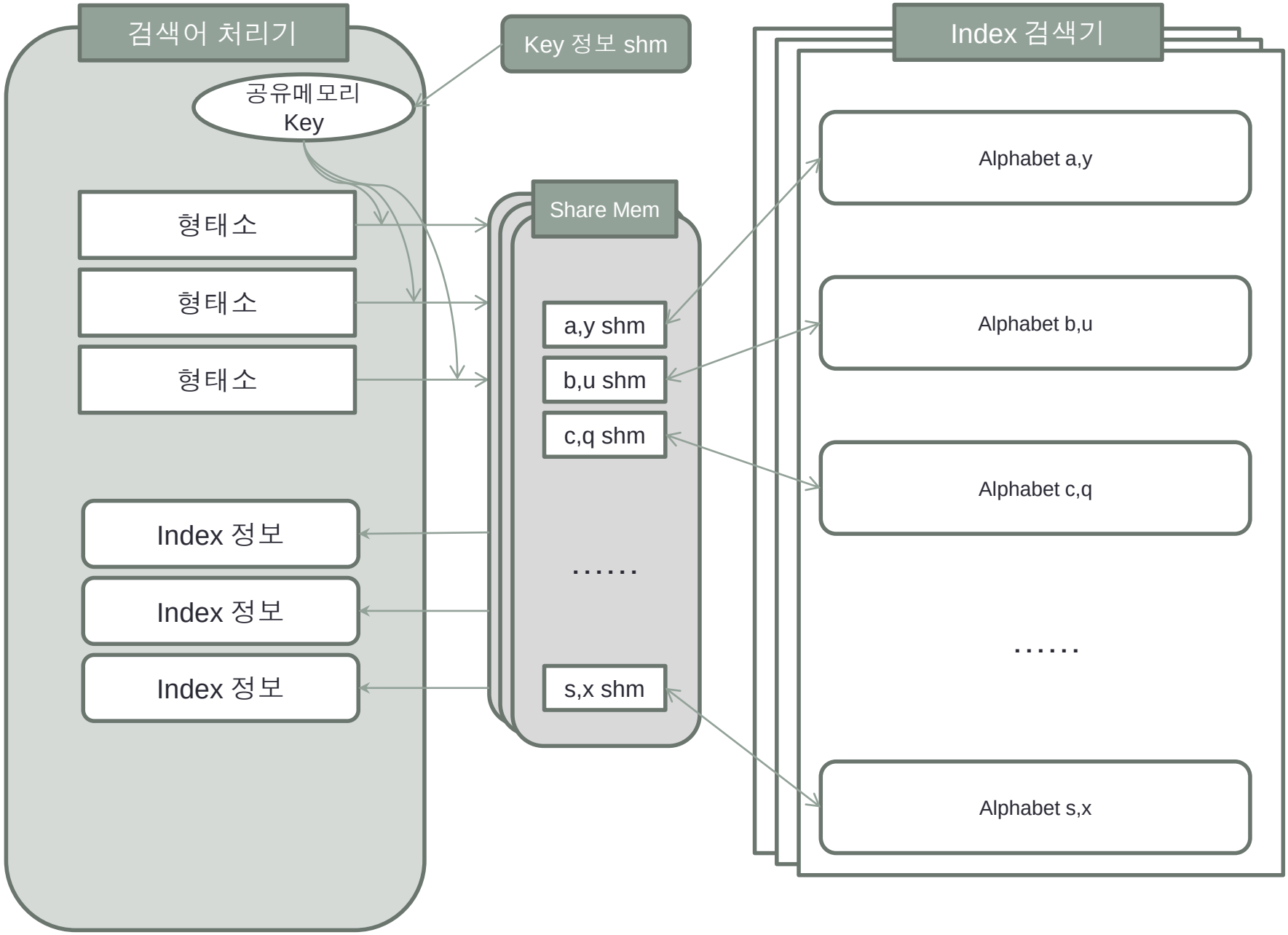
검색이 요청될 때 마다
검색요청저리기 호출

검색어의 형태소 단위로
검색 요청 후 결과 취합

검색어 처리 순서


1.검색어 입력
aqua shoes
2. 검색어 형태소 추출
aqua,
Shoes
6. 조건검색결과 도출
3.aqua 검색요청
5.검색결과 리턴
3.shoes 검색요청
5.검색결과 리턴
Alphabet a,y
4.aqua 색인 정보 검색
Alphabet b,u
Alphabet c,q
.....
Alphabet s,x
4.aqua 색인 정보 검색

검색어 처리기 ⇔ Index 검색기



보완사항

1. 개선 가능 보완사항

a) 색인 정보

- 공유메모리를 활용한 색인 저장

  색인 정보를 공유 메모리로 올려놓고 멀티 스레드로 색인 검색
  자원 소모 절감 및 동시 처리속도 증가

- 색인 저장 시 정렬

  색인을 B-tree로 저장하기 전 해당 색인에 대하여 미리 최신/정확도 순으로
  정렬하여 색인 검색 시 정확한 결과가 먼저 조회되도록 처리

b) 검색 시간 보장

검색시간을 제한하여 시간 내에 검색된 결과만 처리

- 색인 검색 시 분할 검색

  검색어 처리기에서 Index 검색기로 요청 시 결과를 먼저 조회되는(정확도가 높은)
  색인을 받아와 검색 조건 연산에 따른 시간을 단축

  ➔ 예를 들어 각 형태소에 대해 상위 1000건의 색인을 받아와 10건의 결과를
  찾고 만약 10건이 되지 않을 경우 다음 1000건을 받아와 연산.
  보장 시간을 1초라고 가정한다면 0.9초 동안 검색한 결과만을 도출.

고찰

2. 느낀 점
   - 검색어에 대한 모든 결과를 얻기 위한 성능 측정 결과를 바탕으로 장비를 산정하였기 때문에 수 백대의 장비가 산정됨.

   - 어느 정도 만족할 만한 성능을 가정하여 검색 처리 후 장비를 산정한다면 조금 더 현실적일 수 있을 것으로 예상

   - 안정된 서비스 하기 위해 최적화, 효율화 작업이 필요.

   - 좋은 검색 서비스를 제공하려면 개별 검색어 처리에 대한 비용 뿐만 아니라 동시 요청에 대한 구조적인 이해 필요
     ➔ 검색어 처리 시 결과의 정확성/개수 와 자원 사이에서 트레이드 오프를 할 수 밖에 없음.

3. 타 검색엔진과의 경쟁력
   - 멀티태스킹
     엘라스틱 서치, Solr 등의 서비스와 다르게 동시 요청 건 수가 많을 것으로 가정하여 개발하였고 Java vm위에서 동작하는 것이 아닌 Native C언어로 개발된 API로 자원 소모가 적어 동시 요청이 많은 경우에 더 유리함.