LIKE 술어의 질의 수행 비용 모델링†

차명훈*, 성준화**, 박영철***
영진전문대학*, 알티베이스(주)**, 경북대학교***
mhcha@yeungjin.ac.kr, jhseong@altibase.com, ycpark@knu.ac.kr

Modeling of Query Processing Cost for the LIKE Predicate

Myung Hoon Cha*, Joon Hwa Seong**, Young Chul Park***
Yeungjin Junior College*, Altibase**, Kyungpook National University***

요 약

LIKE는 주어진 유형을 찾는 기능을 수행하는 연산자이다. LIKE 술어에 대하여 데이타베이스 관리시스템이 어떠한 탐색 방법을 선택하는가에 따라 질의의 수행 성능에 큰 차이를 가져올 수 있다. 질의 최적화기는 LIKE 연산자를 포함하는 질의를 최적화하는 과정에서 질의 유형과 색인의 존재 여부를 고려하여야 한다. 본 논문은 질의 최적화 과정에서 LIKE 술어를 고려한 질의 수행 비용 모델과 이의 구현 사항을 제시한다.

1. 서 론

데이타베이스 관리 시스템은 질의의 조건을 만족하는 레코드를 찾는 방법으로 모든 레코드들을 차례로 탐색하는 순차 스캔(sequential scan)과 색인을 이용하여 탐색 범위를 제한하는 색인 스캔(index scan)을 제공한다[1]. LIKE는 질의 유형에 부합하는 값을 찾는 연산자로서, LIKE가 포함된 질의를 처리할 경우에는 질의 최적화기가 질의 비용을 계산한 뒤 계산 결과에 따라 순차 스캔을 사용할 것인지 또는 색인 스캔을 사용할 것인지를 결정한다. 질의 최적화기가 질의 비용을 추정하는 과정에서 LIKE 연산자를 포함한 질의에 대하여 적용할 수 있는 비용 모델은 현재까지 알려져 있지 않다.

본 논문은 LIKE 연산자를 포함한 질의의 수행 비용 모델을 제시하며, 바다-II[2, 3]를 기반으로 구현한 결과를 제시한다.

본 논문의 나머지 부분의 구성은 다음과 같다. 제 2장에서는 바다-II의 스캔 관리자가 제공하는 탐색 방법을 제시하고, 제 3장에서는 LIKE 연산자를 포함한 질의를 수행하는데 적용되는 비용 모델을 제시한다. 제 4장에서는 LIKE 연산자의 구현 내용을 소개하고 제 5장에서 본 논문의 결론을 맺는다.

2. 바다-II의 스캔 관리자

바다-II의 저장시스템의 스캔 관리자는 데이타에 대한 탐색 방법으로 순차 스캔과 색인 스캔을 지원한다. 이들은 키 범위(KeyRange), 키 필터(KeyFilter), 데이타 필터(DataFilter)의 세가지 자료형을 제공하며, 범위와 필터는 논리합들의 논리곱 형태, 예를 들어, (id < 50 or name = '홍길동') and (id > 20 or dept = 'cs')와 같은 형식으로 관계 술부를 표현한다.

키 범위는 색인에서 탐색할 키 값의 범위를 나타내며, 키 범위가 나타내는 상한(upper bound) 키 값과 하한(lower bound) 키 값 사이의 색인 값들에 대해서만 탐색하고, 키 범위가 명시되지 않은 경우에는 색인의 전체 범위를 탐색한다.

키 필터는 키 범위를 만족하는 키 값들에 적용할 조건을 의미한다. 키 필터가 명시되지 않은 경우에는 키 범위의 모든 키 값들이 찾기 조건을 만족하며, 키 필터가 명시된 경우에는 키 범위를 만족하면서 키 필터가 명시하는 조건을 만족하는 키 값들만을 탐색한다.

데이타 필터는 순차 스캔과 색인 스캔시 데이타 화일의 데이타 레코드들에 적용할 조건을 나타낸다. 순차 스캔시 데이타 필터가 명시되지 않은 경우에는 모든 레코드들이 탐색조건을 만족하는 것으로 처리하며 데이타 화일의 모든 레코드들을 차례로 탐색한다. 색인 스캔시 데이타 필터가 명시되지 않은 경우에는 키 범위에서 키 필터를 만족하는 모든 레코드들이 탐색조건을 만족하는 것으로 처리하고, 데이타 필터가 명시된 경우에는 키 범위와 키 필터를 만족하면서 데이타 필터가 명시하는 조건을 만족하는 레코드들만을 검색한다.

3. LIKE 술어의 질의 수행 비용 모델

SQL-99 표준[4, 5]은 LIKE 술어를 다음과 같이 정의한다.

<like_predicate> ::= string [not] LIKE pattern [ESCAPE escape_character]

LIKE 연산자의 역할은 질의 대상 문자열인 string에 대하여 pattern이 포함되어 있는지를 검사한 후 string이 pattern을 포함한다면 TRUE를 반환하는 것이다. escape_character는 선택사항으로서 pattern에 있는 메타 문자(meta character : '_', '%' 문자처럼 특수한 의미를 가지는 문자)를 그 문자 자체로 인식하도록 지시하는 데 사용된다.

본 절은 LIKE 연산자를 포함한 질의의 질의 수행 비용 계산 모델을 제시한다. 본 모델은 LIKE 연산자가 포함된 질의의 유형을 분류하고 그 질의의 비용을 측정함으로써 질의 최적화기에 적용할 비용 계산의 지침으로 사용된다.

질의 최적화기가 질의 수행 비용 측정시 고려하는 인자는 질의를 수행할 데이타 화일의 전체 레코드들의 수에 대하여 질의를 만족하는 레코드들의 수의 비율을 나타내는 선택률(selectivity)과 질의 처리를 위해 접근하는 버퍼 페이지들의 수이다.

순차 스캔의 경우에는 데이타 페이지들을 차례로 모두 접근하기 때문에 접근하는 버퍼 페이지의 수는 데이타 화일의 데이타 페이지의 수가 된다. 색인 스캔의 경우에는 색인의 정보와

†본 논문은 한국과학재단 목적기초연구(과제번호 : 2000-2-51200-002-3) 지원으로 수행되었음.

선택률에 따라 접근하는 버퍼 페이지의 수가 달라지므로 질의 유형에 따라 비용 계산 결과를 다르게 도출하여야 한다.

3.1 질의 유형에 따른 선택률 계산

선택률을 계산하기 위하여, LRU 버퍼 페이지 교체 방법을 사용하는 바다-II 환경에서, 한글 문자를 포함한 질의 유형에 대하여 실험하였다. 데이타 화일은 160,000개의 레코드들을 포함하며, 하나의 레코드는 한글 완성형 문자를 균등 분포시킨 3개의 문자로 이루어진 문자열로 구성하였다.

실험결과, 버퍼 페이지 접근 횟수 및 수행 시간의 측면에서 다음과 같이 세가지의 질의 유형으로 구분되었다. 첫번째 질의 유형(유형1)은 '%_%'와 같이 특수문자로만 구성된 경우이다. 두번째 질의 유형(유형2)은 '%종%'의 경우처럼 비특수문자를 한 개 포함하면서 그 위치가 고정되지 않은 경우이다. 세번째 질의 유형(유형3)은 '김%%'처럼 비특수문자를 한 개 포함하면서 그 위치가 고정된 경우이다. 이들에 대한 선택률은 표 1과 같이 도출되었다.

표 1. 한글 문자를 포함한 질의 유형 및 선택률

구분	선택률
유형1	1
유형2	(ceil(160,000/2,350)×3)/160,000 = 207/160,000
유형3	ceil(160,000/2,350)/160,000 = 69/160,000

실험에 사용된 데이타 화일의 한글 문자들이 한글 완성형 코드표에 존재하는 2,350개의 문자들로 균등 분포되어 있기 때문에 160,000개의 레코드들 중에서 2,350개의 문자는 각각 160,000/2,350 = 68.xx개씩 존재할 경우의 수를 가지게 된다. 예를 들어, 질의 유형이 '%종%'일 경우에는 데이타 레코드에서 '종'이라는 문자가 첫번째, 두번째, 또는 세번째에 위치할 수 있으므로 68×3의 개수가 존재할 수 있다. 표 1을 살펴보면 유형2와 유형3의 선택률은 아래와 같은 수식으로 요약된다.

선택률 = (ceil(레코드수/2,350)×자리수)/레코드수

위 수식에서 자리수는 비특수문자가 몇 개의 자리에 올 수 있는지를 나타내는 값이다.

3.2 질의 비용의 계산

질의 처리기의 정당성 검사 모듈에서는 FROM 절에 명시된 스키마 명과 테이블 명을 사용하여 시스템 카탈로그로부터 테이블 정보를 구한 후 메모리에 그 정보를 유지하며, 색인 정보 또한 스키마 명과 테이블 명을 이용하여 테이블의 칼럼에 대한 색인 정보를 구하여 메모리에 유지한다. 표 2는 메모리에 유지되는 정보들 중에서 질의 비용 계산시 사용되는 계산 요소들을 나타낸다.

질의 접근 비용은 레코드 탐색시 접근하는 버퍼 페이지의 수에 대한 예상값이다. 순차 스캔과 색인 스캔시 접근하는 버퍼 페이지의 수는 표 3과 같다.

LIKE 연산자를 포함한 술어의 질의 비용 계산 방법은 다음과 같다. 순차 스캔 비용은 질의 유형에 상관없이 전체 데이타 페이지들의 수가 된다. 색인 스캔의 비용은 질의 유형과 버퍼 접근 횟수를 바탕으로 계산하여야 하며, 질의를 수행하는데 있어 색인의 특성은 버퍼 접근 횟수에 영향을 미치므로 질의 유형의 분류와 함께 고려하여야 한다. 색인을 사용할 경우의 질의 접근 비용의 계산 결과는 표 4와 같다. 표 4의 결과를 고려할 때 색인 스캔을 사용하면 질의 유형에 따라 버퍼 접근 횟수에 큰 차이를 보이는 것을 확인할 수 있다. 특히, 비 클러스터링 색인의 경우에는 그 차이가 가장 크다.

표 2. 질의 비용 계산시 사용되는 계산 요소

	테이블 요소명	의미	예	
데이타 화일	numof_rows	행의 개수	160,000	
	numof_pages	데이타 페이지들의 개수	1,020	
	sizeof_row	하나의 행의 크기	88 bytes	
	ONEPAGE	하나의 데이타 페이지 크기	16KB	
	색인 요소명	의미	예	
			pub_id에 대한 색인	pub_name에 대한 색인
색인 화일	levels	레벨	2	2
	leaf_pages	리프 페이지들의 개수	284	270
	distinct_keys	색인에 속한 별개의 키들	160,000	150,000

표 3. 레코드 탐색시의 접근 비용

구분	비용
순차 스캔 비용	numof_pages
클러스터링 색인이 있을 경우의 색인 스캔 비용	(levels-1)+ ceil(leaf_pages×selectivity)+ ceil(ceil(numof_rows×selectivity)/floor(ONEPAGE/sizeof_row))
비 클러스터링 색인이 있을 경우의 색인 스캔 비용	(levels-1)+ ceil(leaf_pages×selectivity)+ ceil(numof_rows×selectivity)

표 4. LIKE 술어의 질의 유형에 따른 버퍼 접근 횟수

구분	클러스터링 색인	비 클러스터링 색인
유형1	(levels-1)+ leaf_pages+ ceil(numof_pages/ floor(ONEPAGE / sizeof_row))	(levels-1)+ leaf_pages+ numof_pages
유형2	(levels-1)+ leaf_pages+ ceil(numof_pages×selectivity)/floor(ONEPAGE / sizeof_row))	(levels-1)+ leaf_pages+ ceil(numof_pages×selectivity)
유형3	(levels-1)+ ceil(leaf_pages×selectivity)+ ceil((numof_pages×selectivity)/floor(ONEPAGE / sizeof_row))	(levels-1)+ ceil(leaf_pages×selectivity)+ ceil(numof_pages×selectivity)

질의 유형과 버퍼 접근 횟수 사이의 관계를 요약하면 색인이 생성되어 있으면서 질의 유형의 첫 문자가 비특수문자일 경우에는 색인 스캔을 사용하는 것이 효율적이며 이외의 경우에는 순차 스캔을 사용하는 것이 좋다.

3.3 최적 접근 경로의 선택

질의 최적화기는 최적 접근 경로를 선택하기 위하여 질의의 조건절에서 OR로 연결된 절들 간의 비용을 비교한 뒤, 최소의 질의 수행 비용을 가진 OR절을 그 테이블에 대한 최적 접근 경로로 설정하며, 최적 접근 경로로 선정된 OR절을 질의 실행시 먼저 처리하게 된다. 예를 들어, 아래의 그림 1에 제시된

질의를 고려해 보자.

```
SELECT pub_name
FROM publishers
WHERE (pub_id = 1 OR pub_id = 2 OR pub_id = 3) AND
      (pub_name like '김%');
```

그림 1. 견본 질의

그림 1에서 첫번째 OR절(OR절_1이라 하자)은 (pub_id=1 OR pub_id=2 OR pub_id=3)이며, 두번째 OR절(OR절_2라 하자)은 (pub_name like '김%')이다. OR절_1과 OR절_2를 표 2에 주어진 수치값을 적용하여 계산한 뒤 최적 접근 경로를 선정한 결과는 다음과 같다. pub_id에 색인이 존재하고 pub_name에는 색인이 없을 경우에는 OR절_1의 비용은 9, OR절_2의 비용은 1,020이므로 OR절_1을 최적 접근 경로로 선택하게 되며, pub_id에는 색인이 없고 pub_name에만 색인이 있을 경우에는 1,020과 3이 되므로 OR절_2를 최적접근 경로로 선택한다. pub_id와 pub_name에 모두 색인이 있을 경우에는 9와 3이 되므로 OR절_2를 테이블에 대한 최적 접근 경로로 설정하게 된다.

4. LIKE 연산자의 구현

LIKE의 피연산자인 검색어를 색인의 키값 또는 데이타 레코드의 칼럼값과 비교하는 알고리즘은 그림 2와 같이 정규화 단계, 분할 단계, 검색 단계의 세 단계로 수행된다.

```
LIKE_match(OpenFileID, OpenIndexID, ScanMethod, Predicate)
{
  /* 1. 정규화 단계 */
  convert the operand of LIKE operator into
        a simple and equivalent one;

  /* 2. 분할 단계 */
  partition the operand of LIKE operator into
        a list of sub_patterns according to the
        special character '%';

  /* 3. 검색 단계 */
  if (ScanMethod == INDEX_SCAN){
     set KeyRange, KeyFilter and DataFilter from Predicate;
     filtered_records = do_index_scan(OpenFileID, OpenIndexID,
                    KeyRange, KeyFilter, DataFilter);
  }else{
     set DataFilter;
     filtered_records=do_sequential_scan(OpenFileID, DataFilter);
  }
  return filtered_records;
}
```

그림 2. LIKE 연산 알고리즘

정규화(normalization) 단계는 검색어를 정규화한다. 예를 들어 'ab%%d'를 'ab%d'로 변경한다. 분할(partitioning) 단계는 정규화된 검색어 문자열을 특수문자 '%'를 기준으로 SUB_PATT 구조체들의 연결 리스트로 분할한다. 예를 들어, 검색어 '가나%라마_%'에 대하여 '%'를 기준으로 그림 3과 같이 '가나', '라마_', ''의 세 가지 요소로 구성하고, '라마_'와 ''는 '%'로 시작하기 때문에 left_amp 태그를 1로 설정하며, '가나'의 left_amp를 0으로 설정한다. 검색(scanning) 단계는 색인 스캔 또는 순차 스캔을 사용하여 비교 작업을 수행한다.

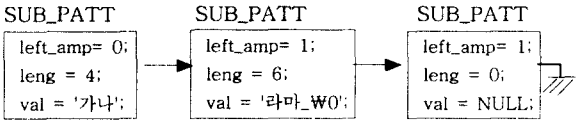

그림 3. SUB_PATT의 연결 리스트로 분할된 상태

예를 들어, 문자열 '가나다라마바사'에 대하여 SUB_PATT 연결 리스트의 각 요소와 대응하는 작업을 다음과 같이 수행한다. 문자열 '가나'와 첫 번째 분할 요소의 값 '가나'가 일치하지만, 문자열 '다라마바'와 두 번째 분할 요소 값인 '라마_'는 일치하지 않는다. 이 경우, 두 번째 분할 요소의 left_amp 태그가 설정되어 있으므로 문자열 '라마바'와 두 번째 요소 값 '라마_'가 일치하는지를 검사한다. 이 경우에는 값이 일치한다. '_'는 임의의 한 개의 문자를 나타내므로 문자열 '사'와 세 번째 요소 값인 ''는 일치하지 않는다. 이 경우, left_amp 태그가 1로 설정되어 있으므로 문자열은 일치하는 것으로 결정한다. 따라서, '가나다라마바사'는 검색 조건을 만족한다.

5. 결론

LIKE 연산자가 포함된 질의는 질의 유형에 따라 색인을 이용하여 질의를 수행하는 경우가 색인을 이용하지 않는 순차 스캔보다 성능이 현저히 향상될 수 있다. 실험 결과, 질의 유형의 첫번째 문자가 비특수문자이면서 색인이 생성되어 있는 경우에는 색인 스캔을 사용하는 것이 효율적이며 이외의 경우에는 순차 스캔을 사용하는 것이 바람직하다는 결론을 내리고 질의 유형에 따른 비용 계산식을 도출하였다. 순차 스캔의 비용은 데이타 레코드 탐색 비용 + 데이타 레코드 참조 비용이며, 클러스터링 색인을 통한 색인 스캔의 경우에는 색인 레코드 탐색 비용 + 데이타 레코드 참조 비용이고, 비 클러스터링 색인을 통한 색인 스캔의 경우에는 색인 레코드 탐색 비용 + 데이타 레코드 참조 비용 + 여분의 디스크 접근 비용(버퍼 적중이 되지 않는 경우)으로 요약된다.

본 논문은 LIKE 연산자를 포함한 질의를 최적화할 경우에 사용할 수 있는 비용 모델을 제시하였으며, 그 비용 모델을 바다-II의 질의 최적화기에 구현한 사항과 LIKE 연산자의 구현 알고리즘을 제시하였다.

참고문헌

[1] J. Gray and A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers, Inc., 1993.

[2] 성준화, LIKE 질의의 최적화를 위한 B^+-트리 연산의 확장과 질의 비용 계산 모델링, 경북대학교 이학석사 학위 논문, 2000년 6월.

[3] 유향미, 단순 술어들로 구성된 조건절의 수행을 위한 질의 최적화기의 개선, 경북대학교 이학석사 학위 논문, 2000년 12월.

[4] J. Melton and A.R. Simon, Understanding the new SQL: A Complete Guide, Morgan Kaufmann Publishers, Inc., 1993.

[5] P. Gulutzan and T. Peltzer, SQL-99 Complete Really, R&D Books Miller Freeman, Inc., 1999.