관계형 데이터베이스에서 효율적인 테이블 튜닝 연산 방법

A Method of Efficient Table Tuning operation in Relational Database

신명주[1], 김용성[2]

Shin Myeongju[1], Kim Yongseong[2]

요약

본 논문에서는 관계형 데이터베이스에서 테이블의 행의 크기, 질의 조건의 속성에 대한 처리 범위, 자료 분포도에 따른 인덱스 효율과 조인 질의시 수행 성능을 평가하여 인덱스 생성 지침을 제시하였다. 이를 위해 질의 선택률과 인덱스 유무를 평가 기준으로 평가하는데 목적이 있다.
실험결과적으로, 인덱스는 테이블의 행의 크기가 클수록 그리고 질의 조건의 속성에 대한 질의 선택률이 많을수록 검색 효율이 떨어졌으며, 단일 테이블에서의 단순 질의와 분할된 테이블에서의 단순 이퀴조인 질의시의 검색 효율을 비교할 때, 질의 조건의 속성에 대한 동등질의나 범위질의 모두 분할된 테이블에서의 단순 이퀴조인 질의 할 때 검색 효율이 떨어졌다. 이러한 실험 결과들을 종합해보면 인덱스는 응용프로그램의 개발 완료 후에 사용된 질의문들을 토대로 인덱스 생성관계, 액세스 경로를 최적화하기 위해서 SQL 튜닝을 수행해야 하며, 이를 통해 데이터베이스의 성능을 향상시키는 과정에서 본 논문에 제시한 인덱스 생성 지침과 활용 지침이 매우 유용한 자료로 활용될 수 있을 것이다.

핵심어: 질의어, 테이블연산, RDB, SQL튜닝, 인덱스

Abstract

In relational database, it is mainly due to inefficient SQL statements that a process occupies the CPU excessively. A processing method for SQL statements with a little data does not cause a serious problem. But in case of the processing with lots of data, it can cause a serious problem in a performance of database system. Though queries produce the same result, the response times of each query can vary depending on the forms of SQL statements. Even though an index is created adequately, it can cause poor performance by using an SQL statement without an index. Also, using unnecessary index brings poor performance of a database system and waste of storage. Therefore, it is important to select and use an index through SQL tuning. In this paper, a test about the efficiency of indexes and the performance of join query with a row size, a process scope of attributes of query condition, and the distribution of data are performed. A guide for index creation in relational database is also proposed.

Keyword: query, table operation, RDB, SQL Tuning, index

1 Department of Computer Center, Cheonbuk Natational Univ, Bakjae-Ro, DuckJin-Gu, JeonJu-City, Cheonbuk-Do 54896, Korea e-mail: silk@jbnu.ac.kr

2 Department of Computer Science & Engreening, Cheonbuk Natational Univ, Bakjae-Ro, DuckJin-Gu, JeonJu-City, Cheonbuk-Do 54896, Korea e-mail: yskim@jbnu.ackr (Corresponding author)
Received(May 31.2016), Review(June 12.2016), Accepted(June 30.2016)

1. 서론

오늘날 최근 데이터베이스 시스템의 성능관리는 응용 시스템의 성능을 좌우하는 중요한 요소가 되어가고 있다. 이를테면 막대한 자금과 시간을 투자하여 구축한 시스템은 초기에는 문제없이 잘 운영되다가도 시간이 지날수록 복잡하고 다양한 사용자의 요구들과 자료의 양의 기하급수적인 증가로 인하여 원하는 정보를 신속하게 제공하기에는 성능상의 한계를 드러내게 된다. 이에 따라, 대부분의 기업 또는 조직은 데이터베이스 시스템의 활용을 필수 요소로 인식하고 있으며, 현재 운용 중이거나 개발할 데이터베이스 시스템의 성능이 전체 응용시스템의 성능에 많은 영향을 미치기 때문에 데이터베이스 시스템의 성능을 향상시키는 일에 많은 노력과 비용을 투자하고 있다[1].

일반적으로 시스템 성능을 향상하기 위해서 현재 관계형 데이터베이스에서 사용되는 SQL(structured query language) 문장을 SQL 튜닝(tuning)함으로써 향상된 성능을 얻기 위한 매우 중요한 부분이다. SQL 튜닝이란 SQL 문장에 대한 튜닝으로 데이터베이스를 효율적으로 사용할 수 있도록 SQL 문장을 수정하고 재작성하는 것이다. 즉, 동일한 결과 값을 갖는 질의라도 SQL 문장으로 질의한다면 데이터베이스 성능은 저하되기 때문이다. 반면에 인덱스를 사용하지 않는 것이 더 효율적인 경우도 있으므로 SQL 튜닝을 통해 인덱스를 적절하게 선택하고 사용하는 것이 중요하다. 데이터베이스의 인덱스를 효과적으로 선택하고 사용하기 위해서는 인덱스 생성 시점, 인덱스를 생성할 속성 선정, 생성된 인덱스를 사용하는 SQL 문장 작성 방법 등을 고려해야 한다[2].

인덱스는 액세스(access)의 효율을 높여주는 것은 틀림이 없지만, 테이블의 특성과 생성시키고자하는 속성의 분포도, 처리범위 등을 정확히 파악하여 지정하여야 한다. 인덱스로 지정된 속성은 그 속성을 사용하는 모든 경우에 영향을 미칠 수 있기 때문에 막연한 추측을 통해 결정해서는 안 되며 가능한 실측자료를 토대로 액세스의 빈도, 처리범위의 크기, 분포도, 테이블 크기, 액세스 유형 등을 감안하여 종합적이고 전략적으로 결정해야 한다[3].

본 논문에서는 데이터베이스 시스템을 효율적으로 사용하는데 있어서 인덱스의 생성 유무와 질의 조건의 선택률에 따른 효율을 평가한다. 그리고 오라클 데이터베이스 관리시스템에서 제공하는 분석도구를 이용한 본 평가를 통해 수행 속도의 성능 개선 방법인 인덱스 설정 방법을 제시하고자 한다.

2. 관련 연구

2.1 SQL 튜닝

데이터베이스에서의 이상적인 튜닝이란 원하는 결과를 가장 빠른 시간 안에 가져오도록 시스템을 조정해 나가는 작업을 의미한다. 결국 튜닝이란 가장 짧은 응답시간을 얻기 위한 데이터베이스 시스템과 개발자 혹은 관리자의 끝없는 도전이며 전쟁이다. 하지만 도전자에겐 한계가 있다. 정해진 시간, 한정된 금액 그리고 정해진 인력으로 맞서 나가야한다. 결국 주어진 자원의 차이 때문에 천편일률적인 튜닝 방법을 적용할 수는 없다. 결국 튜닝이란 "한정된 자원을 가지고 가장 빠른 시간 안에 원하는 결과를 가져오도록 시스템을 조정해 나가는 일련의 작업" 이라는 것이다[3].

그리고 SQL 튜닝은 특정 SQL 질의의 수행 시간을 단축하기 위해 사용자가 취하는 다양한 방법을 통칭한다. SQL 튜닝의 범위는 매우 포괄적인데, 질의 최적화기(optimizer)와 관련한 방법으로는 SQL 재작성, 힌트 사용, 새로운 인덱스 추가, 통계 데이터의 추가/갱신 등을 통해서 질의 최적화기가 더욱 효율적인 실행 계획을 생성하도록 한다[4].

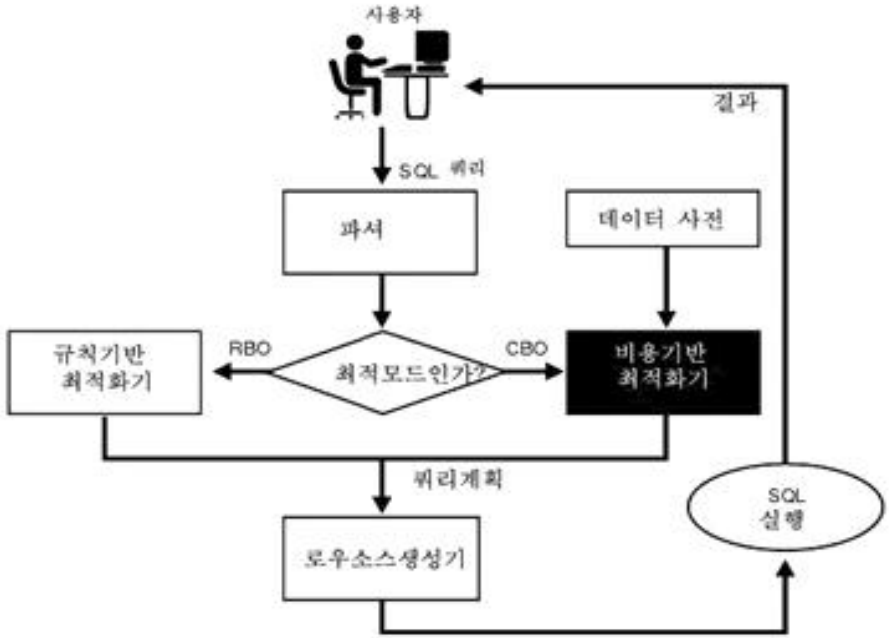
SQL 재작성을 통한 SQL 튜닝은 원래의 SQL 문을 같은 결과를 내지만, 질의 최적화가 더 효과적인 실행 계획을 생성할 수 있는 SQL 문으로 바꾸는 방법으로 힌트사용 하는 방법, 새로운 인덱스 추가 방법, 그리고 통계 데이터의 추가 하고 갱신하는 방법이 있다.

2.2 질의 최적화기

관계형 데이터베이스에서 데이터를 가져오는 방법은 SQL을 사용하는 방법 이외는 없다. 그러나 실행되는 SQL은 데이터베이스 튜닝을 한 상태에 따라, 얻던 물리적인 방법으로 데이터를 저장했는가에 따라, 인덱스를 어떻게 만들었는가에 따라, 데이터를 빨리 가져오도록 어떻게 인덱스를 사용하고 있는가에 따라, 사용자가 어떻게 SQL을 구사했는가에 따라 전혀 다른 결과를 가져올 수 있다. 또한 같은 결과를 가져오는 두 개의 SQL도 내부적으로 데이터베이스 관리시스템의 질의 최적화기가 어떤 방법으로 데이터를 가져오는가에 따라 수행 속도가 전혀 달라질 수도 있다.

질의 최적화기는 사용자가 원하는 데이터를 가져오는 여러 경로와 방법을 알고 있고 SQL을 어떻게 구사하는가에 따라 그때마다 실행 계획을 다르게 세울 수 있으며 이에 따라 데이터를 가져오는 속도에 차이가 나게 된다. 결국은 SQL의 수행속도를 높인다는 것은 사용자가 질의 최적화기를 얼마나 정확히 알고 있는가에 달려 있다.

[그림 2-1]은 질의 최적화기가 SQL을 처리하는 구조이다.



[그림 2-1] Oracle SQL 처리 구조도

[Fig. 2-1] Structure of oracle SQL Process

오라클 질의 최적화기는 SQL 문에 대한 실행 계획을 선택하기 위하여 규칙기반(rule-based)이나 비용기반(cost-based) 접근법을 사용한다. 규칙기반 접근법은 순위가 매겨진 실행 경로를 기반으로 실행계획을 선택한다. SQL 문 실행에 하나 이상의 방법이 있으면, 규칙기반 접근법은 항상 낮은 순위의 실행 경로를 사용한다. 낮은 순위의 실행 경로는 높은 순위 실행 경로의 결합보다 더 빠르게 실행한다. 비용기반 접근법은 질의 최적화기가 모든 가능한 실행 경로와 오브젝트(테이블, 클러스터, 인덱스 등)에 대한 데이터 사전(data dictionary)내의 통계정보와 힌트를 고려한다. 본 논문에서는 질의 최적화 분석에 비용기반 접근법을 사용한다.

2.3 인덱스

질의 최적화기가 최적의 처리경로를 결정하기 위해 사용하는 요소로서 테이블의 행과 하나씩 대응되는 독립적인 객체이다. 생성시킨 속성들과 테이블 행의 논리적 주소로 구성되며 이들간에는 서로 정렬되어 있다. 인덱스는 액세스 효율을 향상시켜 주는 좋은 기능을 가지고 있지만 넓은 범위를 처리해야 하는 경우는 많은 양의 랜덤 액세스를 발생시켜 시스템 과부하의 직접적인 원인이 된다. 물론 결합 인덱스를 잘 활용하면 어느 정도는 개선을 이룰 수가 있으나 그것도 최종적으로 찾고자 하는 범위가 적을 때 가능하다. 처리할 절대 범위가 넓다면 전체 테이블을 읽어야 하는 문제점을 안고 있다[5-7].

인덱스를 구축하여 사용함에 있어서 성능 향상을 기할 수 있는 경우인지를 다각도로 검토할 필요가 있다. 새로 추가된 인덱스는 기존 액세스 경로에 영향을 미칠 수 있으며, 새로 추가된 인덱스가 이미 구현되어 적절히 튜닝 된 프로세스에 나쁜 영향을 미칠 수도 있다. 또한 지나치게 많은 인덱스는 과부하를 발생시킬 수 있으며 튜닝의 효과를 기하기 위해 구축된 인덱스가 지나치게 많을 경우에는 트랜잭션(transaction)시 부담을 줄 수도 있다. 그리고 넓은 범위를 인덱스로 처리할 경우에는 과부하가 발생한다. 질의 최적화기가 최적의 접근경로로 수행할 수 있게 질의 최적화기를 위한 통계 데이터를 주기적으로 갱신시켜 주어야 하며, 인덱스의 개수는 테이블의 사용형태에 따라 적절히 생성한다. 분포도가 양호한 송성도 처리범위에 따라 분포도가 나빠질 수 있다[8].

또한 질의 최적화기가 인덱스를 사용하지 않는 경우는 SQL에 적용하는 인덱스가 테이블에 이미 설정되어 있을 때 설정된 인덱스를 사용하여 데이터를 액세스하는 것이 유리한 경우라 할지라도, 인덱스 속성에 변형이 일어났을 때, 부정형으로 조건을 기술 했을 때, 널(null)로 비교했을 때, 그리고 내부적인 변형이 일어났을 때에는 질의 최적화기의 판단에 따라 인덱스를 사용하지 않는다.

2.4 정규화(normalization)와 반정규화

정규화 작업은 데이터 모델을 검증하는 방법으로 사용되며 이론적으로 5단계의 정규화 작업이 있지만 실제적으로 3단계 까지를 적용하여 사용하는 경우가 대부분이며 3단계 까지 적용하였다

면 정규화 작업은 90%이상 실행되었다고 볼 수 있다. 정규화 작업은 논리적 모델을 검증하는 마지막 단계로 물리적인 설계에 선행되어 작업이 이루어지며 성능을 고려하여 실행되는 비정규화와 반대되는 개념이라 할 수 있다

정규화에 충실하면 데이터가 정확하고 활용성은 향상되나 빈번한 조인 및 집계 데이터 생성 등으로 인해 수행속도가 늦어지는 경우가 발생하기도 한다. 이러한 경우에 수행속도를 빠르게 하기 위해서 정규화를 좀 완화시키는 작업을 하게 된다. 이를 반 정규화라고 한다[9-10].

반 정규화는 그에 따른 장, 단점이 있으므로 사용하기 전에 반드시 인덱스의 조정이나 부분 범위처리 클러스터링 등을 이용하여 해결 할 수 있는지를 철저히 검토한 후 작업을 하는 것이 바람직하며[ ], 주요작업은 테이블 중복화, 테이블 분할과 통합, 속성의 중복함으로써 실행의 효율성을 가질 수 있다.

3. 시험 환경

3.1 시험 데이터 베이스

관계형 데이터베이스 시스템의 성능을 향상시키는 방법은 여러 가지 있으나, 본 논문에서는 응용 프로그램에서 질의할 때 행의 크기, 인덱스 생성관계와 질의 선택률(selectivity)에 따른 효율을 보다 정확하게 평가하기 위해, 실험 데이터는 고유값, 1%, 5%, 20%, 25%, 33%, 50%의 행을 갖는 속성을 만들어 질의 선택 시 고유값, 1%, 5%, 20%, 25%, 33%, 50%의 선택률을 가질 수 있도록 설계하였다.

(1) 스키마

본 논문에서는 질의 선택률을 쉽게 구분하기 위해 속성의 이름에 자료 분포도에 따른 비율을 붙이고, 임의의 값을 갖는 속성에는 일련번호를 사용하였으며, 행의 크기를 조절하기 위해 하나의 속성에는 임의의 문자를 사용하였다. 단일 테이블에서의 질의와 분할된 두 테이블간의 단순 이퀴조인 질의를 실험하기 위한 스키마는 테이블 TAB1과 TAB2는 100byte의 행의 크기를 갖고, 테이블 TAB3는 200byte, 테이블 TAB4는 500byte의 행의 크기를 갖도록 설계하여 객관성을 높이도록 하였다.

(2) 데이터 분포

각 테이블의 COL_SEQ 속성은 순차적인 정수값을 갖는 주 키(primary key)이며, COL_0, COL_1, COL_5, COL_10, COL_20, COL_25, COL_33, COL_50 속성은 각각 고유값, 1%, 5%, 20%, 25%, 33%, 50%의 고유 값 분포(unique value distribution)를 갖는다. 그리고 속성 COL_RANDOM과 COL_61에서 COL_100은 행의 크기를 조정하기 위해서 만든 속성이고 임의의 값을 갖으며 행의 수는 100,000건의 자료를 갖는다. 시험 데이터베이스의 데이터 분포를 정리, 요약하면 [표 3-1]과 같다.

[표 3-1] 시험 데이터베이스의 데이터분포
[Table 3-1] Data distribution of test database

속 성	값	유일한 값 당 튜플의 갯 수	데이터의 분포
COL_SEQ	1,2,.....100,000	1	순차적인 값
COL_0	'AAAA000001',...'AAAA100000'	100,000	고유값 분포
COL_1	'AAAA000001',...'AAAA000100'	1,000	1%의 고유값 분포
COL_5	'AAAA000001',...'AAAA000020'	5,000	5%의 고유값 분포
COL_10	'AAAA000001',...'AAAA000010'	10,000	10%의 고유값 분포
COL_20	'AAAA000001',...'AAAA000005'	20,000	20%의 고유값 분포
COL_25	'AAAA000001',...'AAAA000004'	25,000	25%의 고유값 분포
COL_33	'AAAA000001',...'AAAA000003'	33,000	33%의 고유값 분포
COL_50	'AAAA000001', 'AAAA000002'	50,000	50%의 고유값 분포
COL_RANDOM	임의의 문자열	알수없음	임의의 값 분포

3.2 시험 질의 유형

질의 유형은 행의 크기와 데이터 분포에 따른 인덱스 효율을 파악하는 동등 질의('=')와 범위 질의('between')들로 구성하고, 단순질의와 단순 이퀴조인으로 나누어서 질의한다. 각각의 질의들은 질의 조건의 속성에 인덱스를 생성하지 않았을 때와 비클러스터 인덱스를 생성했을 때 측정한 평균 수행시간을 보고한다. 그리고 실험 1에서 실험 4까지는 각각의 조건에 대한 질의 조건을 제시한다.

실험 1: 단일 테이블에서 동등('=') 질의할 때 질의 조건의 속성에 대한 선택률에 따라 인덱스 효율을 평가한다.

비클러스터 인덱스 사용은 데이터베이스 시스템에서 자료 검색의 효율을 향상스키지만, 검색되는 질의 선택률과 행의 크기에 따라 검색 효율을 떨어뜨릴 수 있다. 실험 1은 행의 크기와 질의 조건의 속성에 대한 데이터 분포에 따른 비클러스터 인덱스 효율을 파악하며, 총 8개의 질의 선택률을 갖는 질의들로 구성하였다.

실험 2: 단일 테이블에서 고유값을 갖는 송성에 범위 질의할 때 질의 선택률에 따라 인덱스 효율을 평가한다.

고유값을 갖는 송성은 비클러스터 인덱스 효율이 높지만 검색되는 질의 선택률과 행의 크기에 따라 검색 효율을 떨어뜨릴 수 있다. 실험 2는 고유 값을 갖는 속성에 범위 질의에 대한 인덱스 효율을 파악하며 총 72개의 질의들로 구성하였다.

실험 3: 두 개의 테이블에서 단순 이퀴조인할 때 동등 질의 조건의 속성에 대한 선택률에 따라 인덱스 효율과 수행성능을 평가한다.

분할된 테이블과 통합된 테이블에서 질의 수행 속도에 미치는 영향은 경우에 따라 엄청난 차이를 가져올 수 있다. 실험 3은 단일 테이블에서의 질의와 분할된 두 테이브란의 조인 질의시 질의 선택률에 따른 인덱스 효율과 수행 성능을 평가한다. 질의는 총 8개로 구성하였다.

실험 4: 두 개의 테이블에서 단순 이퀴조인할 때 질의 조건의 고유값을 갖는 속성에 범위 질의 시 질의 선택률에 따른 인덱스 효율과 수행 성능을 평가한다.

실험 4는 단일 테이블에서의 질의와 분할된 두 테이블간의 조인 질의할 때 고유값을 갖는 속성의 범위 질의에 대한 인덱스의 효율과 수행성능을 파악하며, 총 7개의 질의들로 구성하였다.

4. 실험 결과와 평가

4.1 실험 환경

실험 환경은 128Mb의 주 메모리를 갖는 PentiumⅢ 시스템을 사용하였으며, 운영체제는 Windows 2000을 사용하였고, 데이터베이스는 ORACLE 8.1.7 버전을 사용하였다. 각 질의의 수행 시간을 파악하기 위한 분석도구는 오라클에서 제공하는 SQL_TRACE와 EXPLAIN PLAN 유틸리티를 사용하였다.

각각의 질의들은 작업시간의 정확한 파악을 위하여 데이터베이스를 1회 실행한 후에 질의하고, 재 가동시켜서 다음 질의 작업을 수행하였고, 3번 수행한 후의 평균 수행시간을 보고한다. 각 속성에 대한 인덱스를 생성한 후에는 질의 최적화기가 실행계획을 수립할 때 적절한 통계정보를 사용하도록 분석 작업을 수행하였다.

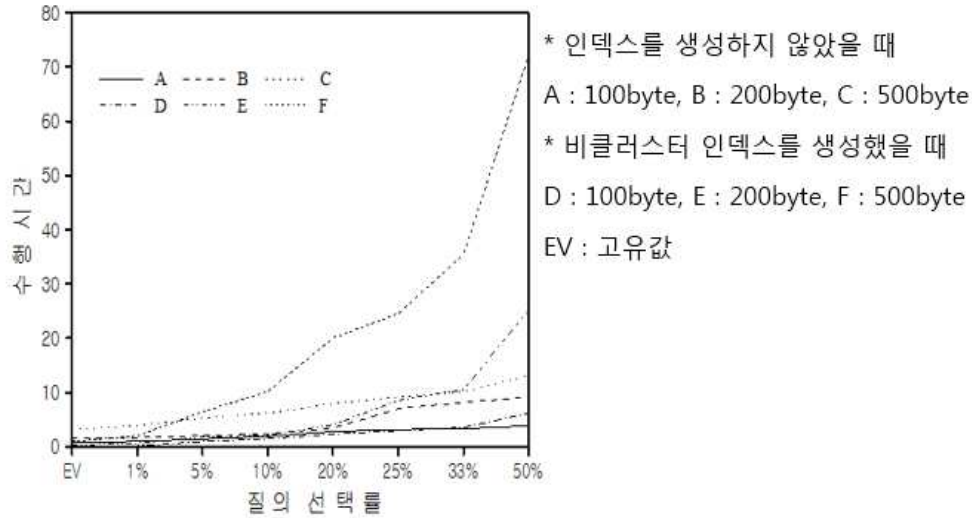
4.2 시험 평가

3장에서 제안한 시험 내용을 실제 실험 환경에 적용하여, 그 효율성을 알아보고 평가한다.

1) 실험 1의 시험 결과와 평가

관계형 데이터베이스 시스템의 동등('=') 질의에 대한 비클러스터 인덱스 효율을 평가한 실험 결과를 나타내며, 100byte 행의 크기에서는 질의 선택률이 25% 이하에서, 200byte 행의 크기에서는 10% 이하에서, 500byte 행의 크기에서는 1% 이하에서 비클러스터 인덱스 효율이 좋았다.

[그림 4-1]에서 알 수 있듯이 행의 크기가 크고 질의 선택률이 많을수록 비클러스 인덱스를 생성했을 때 효율이 떨어지는 폭이 컸으나, 인덱스를 생성하지 않았을 때는 비교적 떨어지는 폭이 완만하였다. 행의 크기가 크고 질의 선택률이 많을 때에는 비클러스터 인덱스를 생성하지 않는 것이 효율적인 방법이다.

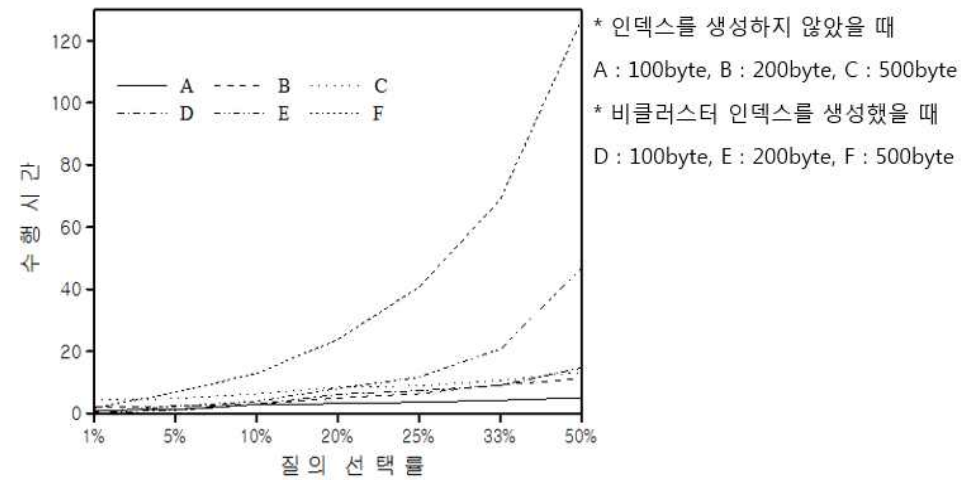


[그림 4-1] 동등 질의에 대한 비클러스터 인덱스 효율평가

[Fig. 4-1] Non cluster Index efficiency estionation of equal query

2) 실험 2의 시험 결과와 평가

관계형 데이터베이스 시스템의 범위질의('between')에 대한 비클러스터 인덱스 효율을 평가한 시험결과를 나타내며, 100byte 행의 크기에서는 질의 선택률이 10% 이하에서, 200byte 행의 크기에서는 5% 이하에서, 500byte 행의 크기에서는 1% 이하에서 비클러스터 인덱스 효율이 좋았다.



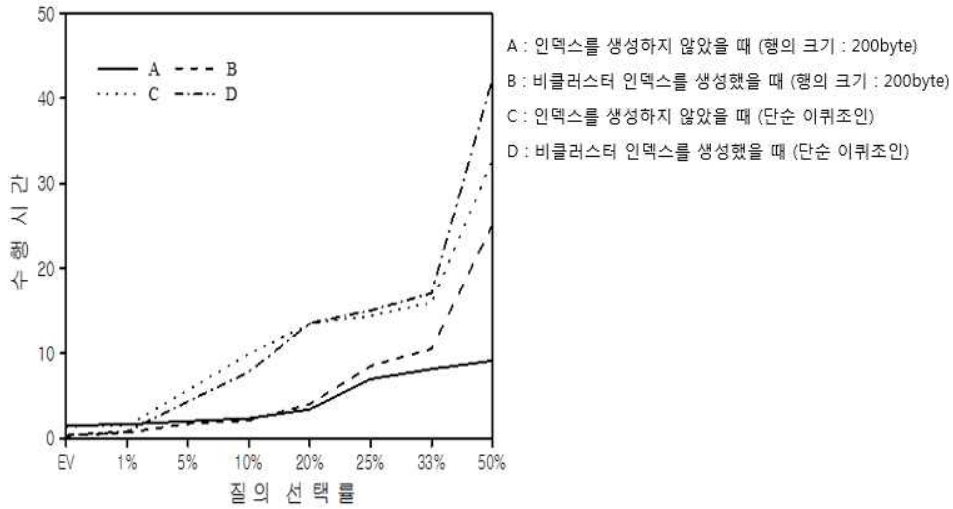
[그림 4-2] 범위 질의에 대한 비클러스터 인덱스 효율평가

[Fig. 4-2] Non Cluster Index efficiency estimation of between query

[그림 4-2]에서는 행의 크기가 크고 질의 선택률이 많을수록 비클러스터 인덱스를 생성했을 때 효율이 떨어지는 폭이 동등 질의보다 컸으며, 인덱스를 생성하지 않았을 때는 떨어지는 폭이 비교적 완만하였다. 결과적으로 행의 크기가 크고 질의 선택률이 많을 때에는 비클러스터 인덱스가 생성되었어도 비클러스터 인덱스 속성을 변형하여 인덱스를 사용하지 않게 하는 것이 효율적인 방법이다.

3) 실험 3의 시험 결과와 평가

관계형 데이터베이스 시스템에서 두 테이블간의 단순 이퀴조인 할 때 비클러스터 인덱스에 대한 효율과 수행성능을 평가한 시험결과를 나타낸다.

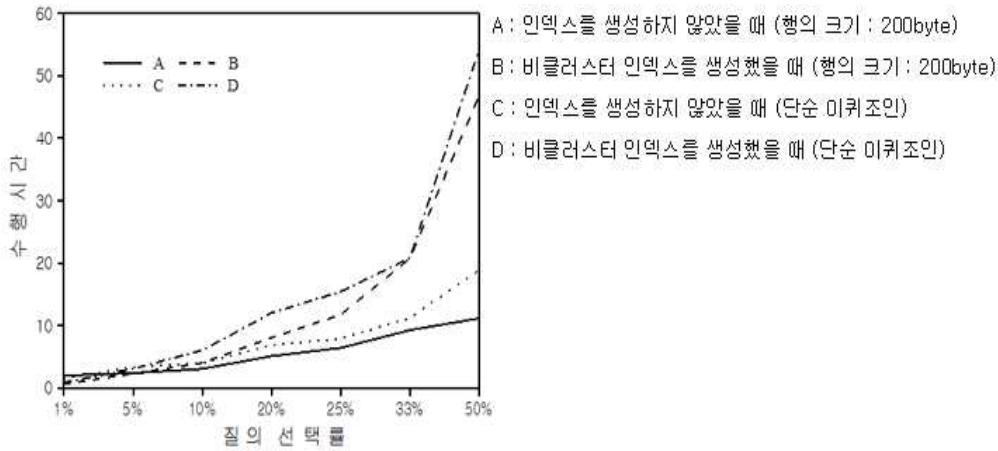


[그림 4-3] 동등 질의시 질의 선택률에 따른 속도변화
(단순 이퀴조인과 행의 크기: 200byte)
[Fig. 4-3] Velocity variation of query rate for equal query

[그림 4-3]은 100byte 행의 크기를 가진 두 테이블에서의 조인 질의 시 질의 조건의 속성에 대한 선택률과 200byte 행의 크기를 가진 테이블에서의 속성에 대한 선택률과 수행 속도를 비교하였을 때 1% 이하는 수행속도에 차이가 없으나 질의 선택률이 5% 이상일 때는 조인 질의 시 전반적으로 수행 속도가 떨어졌다.

4) 실험 4의 시험 결과와 평가

관계형 데이터베이스 시스템에서 두 테이블간의 단순 이퀴조인 시 질의 조건의 속성에 고유값을 갖는 속성의 범위 질의에 대한 비클러스터 인덱스 효율과 수행성능을 평가한 시험 결과를 나타낸다.



[그림 4-4] 범위 질의시 질의 선택률에 따른 속도변화
(단순 이퀴조인과 행의 크기: 200byte)
[Fig 4-4] Velocity Variation of query rate for between query

[그림 4-4]는 100byte 행의 크기를 가진 두 테이블에서의 조인 질의시 질의 조건의 속성에 대한 범위 질의 선택률과 200byte 행의 크기를 가진 테이블에서의 질의 조건의 속성에 대한 범위 질의 선택률과의 수행 속도를 비교한 경우로 조인 질의시가 전반적으로 수행 속도고 떨어졌다. 경우에 다라 분할된 테이블은 오히려 수행 속도를 나쁘게 하므로 신중한 결정이 요구된다.

5. 결론 및 향후 연구과제

데이터베이스 시스템에서 프로세스가 과도하게 CPU를 점유하는 경우는 주로 비효율적인 SQL이 원인이 경우가 많다. 일반적으로 응용프로그램 작성 시 소량의 데이터를 처리했을 때는 그 처리 방법이 아무리 비효율적이라 하더라도 바로 원하는 결과를 얻을 수 있어 문제의 심각성을 느끼지는 못하지만, 대량의 데이터에서는 성능상의 한계를 가진다.

따라서 본 논문에서는 테이블의 행의 크기, 질의 조건의 속성에 대한 처리 범위, 자료 분포도에 따른 인덱스 효율과 조인 질의 시 수행 성능을 평가하여 인덱스 생성 지침을 제시하였다. 이를 위해 질의 선택률과 인덱스 유무를 평가 기준으로 하여 실험 1에서 실험 4까지의 평가를 진행하였다.

실험결과 얻은 주요결론은, 인덱스는 테이블의 행의 크기가 클수록 그리고 질의 조건의 속성에 대한 질의 선택률이 많을수록 검색 효율이 떨어졌으며, 단일 테이블에서의 단순 질의와 분할된 테이블에서의 단순 이퀴조인 질의시의 검색 효율을 비교할 때, 질의 조건의 속성에 대한 동등질의나 범위질의 모두 분할된 테이블에서의 단순 이퀴조인 질의 할 때 검색 효율이 떨어졌다.

이러한 실험 결과들을 종합해 보면, 인덱스는 응용프로그램의 개발 완료 후에 사용된 질의 문들을 토대로 인덱스 생성관계, 액세스 경로를 최적화하기 위해서 SQL 튜닝을 수행해야 하며, 이를 통해 데이터베이스의 성능을 향상시키는 과정에서 본 논문에 제시한 인덱스 생성 지침과 활용 지침이 매우 유용한 자료로 활용될 수 있다.

향후 연구 과제로서는 관계형 데이터베이스에서 필수적으로 사용되고 잘못 사용되었을 때는 엄청난 수행 속도를 떨어뜨리는 조인 연산의 효율적인 처리를 위한 연구가 필요하다.

ISSN: 2508—9099 JDAEM

References

[1] Y. L Choi, B. K. Yoon, K. W. Chong, The Juurnal of Korean Institute of CALS/EC. (2002), Vol.7, No.1, pp.108-127.

[2] Oracle 8 Database Administration, Performance Tuning Workshops, SQL Tuning, ORACLE, 1998.

[3] ORACLE Inc. http://otn.oracle.co.kr/docs/Oracle817/server.817/a76992/optimops.htm#30169, 2000.

[4] S. W. Lee, "Oracle Optimizer Basic Principles", Oracle, oracle.com/kr, p.14, 2003.

[5] Y. W Kim, Master's Thesis, Incheon Univ. (2001).

[6] S. Ress, iBM Tech & Report. (2013).

[7] http://kb.informatie.com/h21/Howto%20Library/110896

[8] K. P. Devarakonda, sqoop Performance Turning Guidelines, informotia. (2015).

[9] ORACLE, Oracle Database Performance Tuning Guide 118 Release2. (2014).