어플리케이션 튜닝을 통한 데이터베이스 시스템 성능 향상

이병헌[0] 최용락 정기원*
숭실대학교 컴퓨터학과, 숭실대학교 컴퓨터학부*
leebh42@info.soongsil.ac.kr, ylch@hitel.net, chong@computing.soongsil.ac.kr

Application Tuning For Increased Database System Performance

ByoungHeon Lee[0] YongLak Choi Kiwon Chong*
Dept. of Computing, Soongsil Univ. , School of Computing, Soongsil Univ.

요 약

기존 시스템에서 보다 정확하고 신속한 정보의 제공에 대한 사용자들의 요구사항을 충족시키기 위해서는 시스템의 개선이나 현재 운영중인 시스템의 조율이 필수적으로 여겨지고 있다. 시스템 개발자는 관계형 데이터베이스 시스템의 성능을 저하 시키는 요인 중 응용 프로그램들과 저장 프로시저, 트리거, 패키지, 뷰 등에 대한 성능 향상을 위하여 튜닝을 고려 하여야만 한다. 특히, 응용 어플리케이션 수행에 많은 시간을 소요하거나, 많은 자원을 필요로 하는 응용 프로그램들을 중점적으로 분석하여 적절한 튜닝을 수행한다. 또한, 오라클에서 제공하는 도구들을 이용하여 자료나 질의문의 특성을 파악한 후에 효과적인 개선을 통하여 데이터베이스 시스템의 성능 및 효율을 높이는 방법에 관심을 두고 있다.

1. 서론

데이터베이스 관리 시스템(DBMS: DataBase Management System)이란 데이터베이스를 사용하는 사용자의 요구사항을 수행할 수 있는 많은 기능과 다양한 구조를 갖고 있으며 이 구조들 사이를 연결하고, DBMS 기능들과 구조를 이용하는 데이터베이스 언어로 구성된 일종의 소프트웨어 시스템이다. 데이터베이스 관리 시스템을 사용함으로써 데이터 중복 최소화, 일관성 유지, 무결성 보장 등의 장점을 얻을 수 있으며 대부분의 데이터 관리 시스템이 사용하는 관계형 모델은 효과적인 구현을 위해 단순하고, 이식성이 높으며, 표현력이 뛰어난 SQL 언어를 지원한다.[1] 이러한 관계형 모델이 가지는 여러 장점들에 부응하여 현존하는 정보 시스템들은 대부분 관계형 데이터베이스 시스템으로 재구성 되어져 왔다. 하지만, 적절하게 설계된 데이터 모델을 적용한 시스템이라고 할지라도 기하 급수적으로 늘어나는 데이터와 빠르게 진행되는 업무의 변화 등에 따라서 그 성능을 제대로 발휘 할 수 없는 것이 현실이다. 관계형 데이터베이스 시스템의 관리는 정확하고 신속한 정보의 제공을 위하여 필수사항이 되어져 왔다. 본 논문에서는 관계형 데이터베이스 시스템을 조율하기 위한 방법 중 특히 어플리케이션과 관련된 부분을 연구하고, 이를 통하여 효율적인 시스템을 구축하고 사용할 수 있도록 한다. 또한, 관계형 데이터베이스 시스템 중 가장 최초로 상용화 되어 널리 사용되고 있는 오라클의 예를 들어 설명을 돕고자 한다.

2. 개요

시스템 관리자는 데이터베이스 응용, 데이터베이스 자체, 그리고 운영체제의 조정 등을 통하여 데이터베이스 시스템 성능을 향상 시킬 수 있다.[2] 이러한 조정 과정을 튜닝이라고 한다. 따라서, 시스템 관리자는 적절한 튜닝을 통하여 특정 응용시스템이나 하드웨어에 대하여 데이터베이스 성능을 최대로 향상 시킬 수 있다. 관계형 데이터베이스 시스템의 성능 저하에 영향을 미치는 요인들은 그림[1]에서 보여지는 바와 같다.

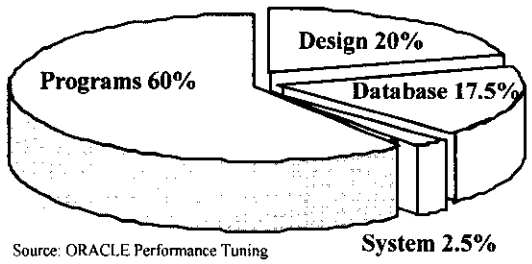
시스템 관리자는 시스템의 성능향상을 위하여 각각의 분야의 특성에 맞추어 튜닝을 고려하여야만 한다.[2] 우선, 시스템 튜닝을 위하여서 메인 메모리에 적당한 크기의 버퍼를 설정하고, 불필요한 I/O를 줄이거나, 충분한 메모리 자원을 확보하여 페이징이나 스와핑의 발생을 최소화 시켜야 한다. 또한, 정규화 튜닝이나 성능향상을 위한 비정규화 튜닝, 그리고 인덱스의 사용 또는 클러스터링 된 테이블의 이점 등은 디자인 측면에서 고려되어야 할 사항들이다.

3. 튜닝의 기본 원칙

성능 향상을 위하여 기본적으로 고려하여야 한 사항들은 다음과 같다.[3]

1) 전체를 대상으로 생각하고 해결은 부분적, 국부적으로 수행

전체 튜닝의 일반적인 접근 방식은 프로세서의 활용에 대한 내용이나 입출력 활동에 대한 내용, 또는 페이징 방법 등에 대한 하드웨어적인 접근 방법이 먼저 고려된다. 또한, 빈번하게 발생하는 특정 질의에 대해서는 인덱스를 생성, 데이터와 로그를 서로 다른 디스크로 분리하고, 국부적인 특정 질의에 문제의 해결은 그 질의가 매우 중요할 경우에 수행하여 본다



그림[1] 시스템 성능에 영향을 미치는 요인들

2) 병목 현상을 분할

과부하가 발생하는 시스템 구성요소에 대한 속도 개선의 고려를 위하여 인덱스의 추가를 통한 방법이나, 기존 질의의 재구성 등을 고려해 볼 수 있다. 또한, 공간적 부분화, 논리적 자원의 부분화, 시간적 부분화 등을 통하여 부하가 집중적으로 걸리는 자원을 나누어 사용하거나 부하가 걸리는 시간을 나누어서 존재하는 시스템의 구성 요소들의 집중적 부하를 줄인다.

3) 높은 시스템 가동 비용과 낮은 운영 비용의 고려

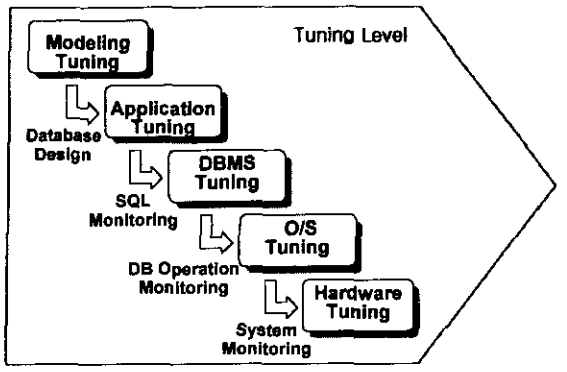
디스크상에서 자료를 읽을 경우의 가장 첫 번째 바이트를 빨리 찾는 방법이나 분산 시스템에서의 전달 내용의 첫 번째 바이트를 빨리 전송하는 방법 혹은, 단순 질의의 문법적 해석의 성능 개선을 위하여 미리 컴파일 된 실행 질의로 변환하거나 데이터베이스 접근 시 프로그래밍 언어 사용 등을 고려한다

4) 서버 작업과 클라이언트 작업을 파악

클라이언트와 서버에 관계된 자원들의 사용량을 계산하여 서버 과부하시 클라이언트로 작업을 분배하거나, 정보의 적절한 배치 여부나 데이터베이스 작업과 대화형 작업의 관계 여부 등을 고려하여 데이터베이스 시스템과 응용프로그램 사이의 작업을 할당 한다.

4. 튜닝의 목적과 단계

튜닝의 목적은 튜닝 방법론 절차에 따라 현행 정보 시스템에 존재하는 문제점을 분석하고, 분석된 문제점을 단계적으로 해결함으로써 자원활용을 극대화 하여 정보 시스템을 안정 시킴으로써 사용자 만족과 관리력을 향상 시킨다. 튜닝은 그림[2] 와 같이 단계화 되어질 수 있고, 각 단계에서 고려되어져야 할 사항이 있다. 그림[1]에서와 같이 프로그램 측면에서 시스템 성능을 저하시키는 부분은 그림[2]에서 제시한 튜닝의 단계 중 어플리케이션 튜닝의 단계에서 고려 되어질 수 있다.

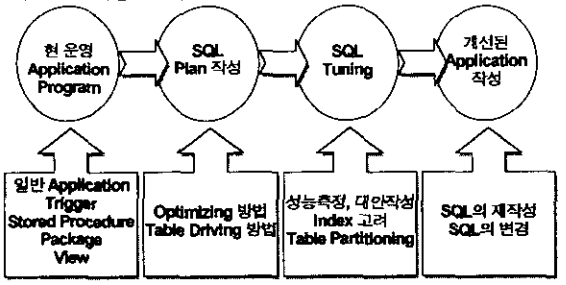


그림[2] 튜닝의 단계

5. 어플리케이션 튜닝

어플리케이션 튜닝은 현재 운영중인 모든 응용 프로그램을 대상으로, 즉 트리거, 저장 프로시저, 패키지, 뷰 등을 포함하여 집중적인 부하가 발생하는 시점의 해당 프로그램 수행에 필요한 시간 및 필요로 하는 데이터들을 조사한다. 그리고, 튜닝을 필요로 하는 대상을 선정하여 선정된 어플리케이션을 위주로 집중적인 튜닝을 실행한다. 튜닝의 방법으로는 케이스 도구(CASE TOOL)를 이용하여 해당 프로그램의 운영 계획을 작성하고 작성된 계획이 최적의 성능을 나타낼 수 있는 운영인가를 파악하여 처리 되는 순서를 변경하거나 인덱스 등을 추가한다. 또한 최적화가 적절히 적용되는가 등을 파악하여 이를 조절한다. 또한 어플리케이션 튜닝은 현재 운용중인 데이터베이스와 관련된 모든 응용 프로그램을 대상으로 하며 집중적인 부하가 발생하는 시간에 특히 영향을 주는 응용 프로그램들 또는 활용도가 높은 응용 프로그램 위주로 진행한다. 그리고, 운용되는 응용 프로그램의 통계적 자료 등을 모두 활용하여 데이터 사용량 분석을 수행하여 집중적 부하가 발생되는 데이터를 파악한다. 이렇게 파악된 프로그램들과 집중적인 부하를 발생하는 데이터들은 튜닝의 대상이 되는 것이다.

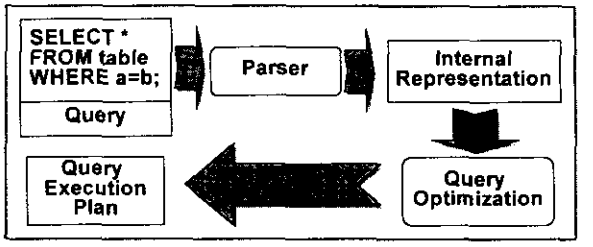
1) 어플리케이션의 튜닝 순서
   ① 현재 사용중인 모든 어플리케이션의 리스트 데이터를 작성
   ② 특정 시간대 별로 프로그램의 수행시간 파악
   ③ 튜닝 대상 프로그램 설정
   ④ 프로그램 분석
   ⑤ 데이터 사용량 분석
   ⑥ 분석자료 검토 및 변경지침 작성
   ⑦ 데이터 처리 방법 변경을 포함한 프로그램 수정
   ⑧ 데이터 분산
   ⑨ 시간대별 운용 방법의 변경
2) 어플리케이션 튜닝의 단계
개략적인 어플리케이션 튜닝의 단계는 그림[3]과 같다.



그림[3] 어플리케이션 튜닝의 단계

어플리케이션을 위한 튜닝 중 패키지, 프로시저, 트리거와 같은 각 유형의 PL/SQL 객체가 오라클 내에 컴파일된 형태로 저장되어 있다는 사실 때문에 튜닝할 필요가 없는 것은 아니다.[2] 이러한 객체들은 구문이 분석된 다음, 데이터 딕셔너리에 저장되어 있으므로 공유 풀에 캐쉬 되어 구문 분석된 버전이 재사용될 수 있어 실행 속도를 증가 시킬 수 있는 장점이 있다. 그러나, 공유풀에 오래 상주하였지만 사용되지 않아서 삭제된 객체가 실행될 때 다시 구문 분석을 해야 되기 때문에 성능에 많은 영향을 준다. 이 때 사용 어플리케이션의 분석을 통하여 자주 사용되거나 중요도가 높은 객체를 공유 풀에 고정할 수 있는데, 이것은 오래되어도 삭제되지 않도록 보장하기 때문에 자원 비용이 많이 드는 구문 분석 작업을 최소화 할 수 있다. 시스템 관리자는 그러한 PL/SQL 프로시저의 사용을 추적하여 공유되는 오라클 구조를 효과적으로 사용할 수 있도록 보장 할 수 있다. 이러한 어플리케이션 추적과 자원 분석을 위해서 시스템 관리자는 오라클에서의 어플리케이션을 등록하기 위한 방법을 이해해야 한다. 오라클은 데이터베이스와 관련된 어플리케이션을 등록하기 위해서는 DBMS_APPLICATION_INFO 패키지 내에 있는 프로시저를 사용하고, 그것의 이름과 실행 내용은 V$SESSION과 V$SQLAREA 뷰에 기록된다. 시스템 관리자는 이러한 내용을 가지고 현재 시스템에 등록되어 있는 모듈을 추적하여 시스템 튜닝에 필요한 자료를 확보 할 수 있다. 또한 모듈에 의해 사용되는 자원의 양을 추적하기 위하여 이러한 정보를 이용할 수 있다.

질의 처리 작업에서 가장 힘든 부분은 질의 최적화 이다. 질의 최적화는 질의 처리를 위해 효과적인 질의 계획이나 저장 시스템에 대한 요청 순서를 선택하는 것이다.[1] 일반적으로 질의는 그림[4]과 같은 방법으로 처리되고 최적화 된다.



그림[4] 질의문 처리

비효율적인 질의문이란 옵티마이저가 인덱스를 사용할 수 없는 경우, 혹은 사용된 문장에 트리 구조나 그룹 함수 혹은 DISTINCT와 같은 키워드가 포함되어 있거나, 문장이 복합 뷰를 기본으로 하여 질의를 수행한다. 이렇게 성능을 저하 시키는 질의문을 찾아 수정을 하여야 한다. 어플리케이션 실행 시 질의문이 적절하게 작성되지 않았을 경우, 데이터가 선택되는 방법이나 선택된 데이터의 양은 어플리케이션의 성능에 심각한 영향을 미친다. 표[1]는 롯데 쇼핑 통합 영업 정보 시스템의 데이터베이스 모델링 검증 및 튜닝 작업 프로젝트의 튜닝 결과 보고서 중 '사후 매가 집계표'에 대하여 단순히 테이블 처리 순서(Table Driving)를 재배치 하여 얻어진 성능 향상에 대한 예를 보였다. 기존 데이터베이스 시스템에 대한 자료나 구문 분석은 오라클에서 제공하는 도구들을 이용하였다. 이러한 단순한 테이블 처리 순서 재배치를 통해서 약 15배 정도의 성능향상을 얻어 낼 수 있었다.

(튜닝 수행 전)

```
SELECT B.GWA, A.PUMBUN_CD,
A.MARGIN_RATE, SUM(A.MAEIP_WONGA),
...(중략)
, SUM(A.S_BANPUM_MAEGA)
FROM GIA001 A, RMD_PUMJOJIKHIST B
WHERE A.STR_CD='0001' AND
(A.GUMPUM_DATE BETWEEN '19981106' AND '20001207')
AND A.PUMBUN_CD LIKE '025125'||'%'
AND B.JOJIK_GB='0' AND B.GWA LIKE '00'||'%'
AND B.KYE LIKE 'A'||'%' AND
B.GWA||A.PUMBUN_CD||TO_CHAR(A.MARGIN_RATE)>'20'
AND A.GUMPUM_DATE BETWEEN B.START_DATE AND
B.END_DATE
AND A.PUMBUN_CD=B.PUMBUN_CD
AND A.STR_CD=B.STR_CD
GROUP BY A.GUMPUM_DATE, B.GWA,
A.PUMBUN_CD, A.MARGIN_RATE
```

(튜닝 수행 후)

```
SELECT B.GWA, A.PUMBUN_CD,
A.MARGIN_RATE,
...(중략)
,SUM(A.S_BANPUM_MAEGA)
FROM RMD_PUMJOJIKHIST B , GIA001 A
WHERE A.STR_CD='0001' AND
(A.GUMPUM_DATE BETWEEN '19981106' AND '20001207')
AND A.PUMBUN_CD LIKE '025125'||'%'
AND B.JOJIK_GB='0' AND B.GWA LIKE '00'||'%'
AND B.KYE LIKE 'A'||'%' AND
B.GWA||A.PUMBUN_CD||TO_CHAR(A.MARGIN_RATE)>'20'
AND A.GUMPUM_DATE BETWEEN B.START_DATE AND
B.END_DATE
AND A.PUMBUN_CD=B.PUMBUN_CD
AND A.STR_CD=B.STR_CD
GROUP BY A.GUMPUM_DATE, B.GWA, A.PUMBUN_CD,
A.MARGIN_RATE
```

그림[5] 질의문 튜닝 예제

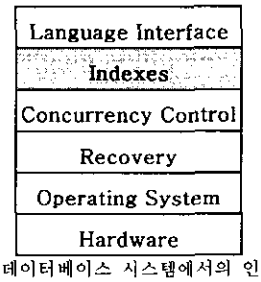
어플리케이션 개발자들은 질의문의 튜닝을 위하여 다음의 단계를 고려 해야만 하고, 오라클에서 제공되어지는 성능 분석 도구를 이용하여 각 단계별 결과를 검사하여 튜닝에 필요한 작업을 한다.[2]
① 스키마 튜닝을 한다.
   - 필요하면 인덱스를 추가하거나 클러스터를 생성한다.
② 질의문장 최적화의 재사용성에 대해 설계한다.
③ 질의문장을 설계하고 튜닝 한다.

④ 옵티마이저로 최대 성능을 달성한다.

Attribute	구분	Last Value	Avg. Value
Total CPU (Sec.)	튜닝 전	0.15	0.15
	튜닝 후	0.01	0.01
Elapsed (Sec.)	튜닝 전	0.16	0.16
	튜닝 후	0.01	0.01

표[1] 튜닝 예제의 질의 수행 결과

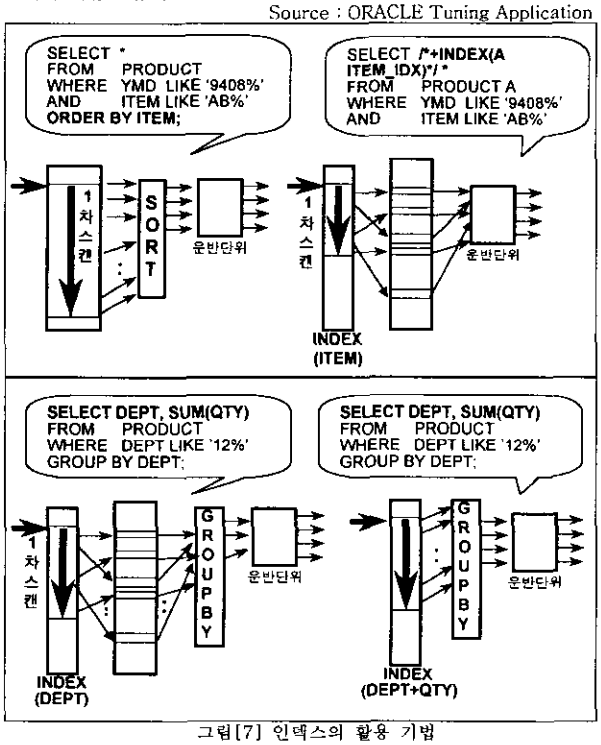
인덱스는 데이터베이스 내부의 테이블의 행(Record)에 대하여 질의를 할 경우에 속도를 높이기 위하여 사용된다. 일반적으로 관계형 데이터베이스에서 사용하는 인덱스의 종류에는 주 식별자 인덱스(Primary Key Index), 부 식별자 인덱스(Alternate Key Index), 외부 식별자 인덱스(Foreign Key Index)로 분류된다. 오라클은 외부 식별자에 대한 인덱스를 제외하고, 무결성 제약(Integrity Constraints)으로 정의한 모든 컬럼에 대하여 인덱스를 자동으로 생성한다. 데이터베이스 시스템 구조에서의 인덱스의 위치는 그림[6]와 같다.[3]



그림[6] 데이터베이스 시스템에서의 인덱스 위치

인덱스의 설정, 관리, 분산 및 설정된 인덱스를 활용하는 방법에 따라 데이터베이스 성능을 향상 시킬 수 있다.[4] 그림[7]은 단순히 테이블을 검색하여 데이터를 정렬할 경우와 인덱스를 사용할 경우 처리 방법에 따른 성능향상에 대한 사례와 검색하는 데이터의 종류에 따라서 질의문의 복성에 맞게 생성된 인덱스가 어떻게 성능에 도움을 주는지의 예를 보였다.

Source : ORACLE Tuning Application



그림[7] 인덱스의 활용 기법

하지만, 인덱스의 사용은 인덱스 구조에 따라서 데이터의 작업에 속도의 저하를 발생 시키고, 인덱스를 유지, 관리하기 위해 사용되는 자원의 부적절하거나 불필요한 낭비 등을 고려하여 설정되어야 한다. 또한, 질의문에 사용하지 않는 인덱스의 설정 역시 입력, 수정, 삭제에 많은 부담을 준다. 이러한 인덱스를 찾아 제거 하는 것은 시스템 성능 향상에 많은 도움을 준다. 표[2]는 인덱스의 사용 여부에 따른 성능 차이를 보여 주고 있다.

Source : ORACLE SQL Tuning

Attribute	Index 사용	Value	Detail
Total CPU (Sec.)	No	62.19	Parse(0.43) Execute(0.00) Fetch(61.76)
	Yes	0.46	Parse(0.44) Execute(0.00) Fetch(0.02)
Elapsed (Sec.)	No	79.90	Parse(0.54) Execute(0.00) Fetch(79.36)
	Yes	0.88	Parse(0.66) Execute(0.00) Fetch(0.22)

표[2] 인덱스 사용여부에 따른 성능 변화

인덱스와 더불어 스키마 튜닝에서 고려해야 될 부분인 클러스터링은 I/O성능을 개선하고, 검색시간을 개선하며 스토리지 오버헤드를 줄일 수 있는 테이블을 저장하는 대체 방법이다.[1] 클러스터에는 해쉬 클러스터, 인덱스 클러스터의 두 가지의 형태가 있다.

그림[4]에서와 같이 질의문은 먼저 질의문을 해석 한 후 옵티마이저에 의하여 최적화된 질의문 실행 계획을 작성하여 처리한다. 옵티마이저는 규칙기반(Rule-based) 접근법이나, 비용기반(Cost-based) 접근법을 사용하여 질의문에 대한 최적화된 실행 계획을 결정한다. 시스템 관리자는 EXPLAIN PLAN으로 옵티마이저가 질의문을 위해 선택한 실행 계획에 대한 정보를 얻어 낼 수 있고, 오라클에서 제공하는 성능 측정 도구 중 어플리케이션 튜닝에 특별히 도움이 되는 SQLTRACE를 사용하여 튜닝에 필요한 정보를 수집할 수 있다. 그리하여, 시스템 관리자는 이러한 실행계획을 점검하여 질의문에 대한 비효율이 발생한 원인을 찾을 수 있고 좋은 실행 계획이 수립될 수 있도록 하는 방법을 찾을 수 있다. 이러한 실행계획을 자주 참조하는 것이 바로 옵티마이저를 이해할 수 있는 최선책이며, 질의 문장을 튜닝할 수 있는 능력을 키울 수 있는 가장 좋은 방법이다. 물론, 튜닝을 위하여 다른 종류의 여러 가지 도구들이 존재한다. 하지만, 오라클에서 일반적으로 사용되고 있는 도구들을 본 연구에서 소개하였다.

어플리케이션 튜닝을 위해서 데이터 설계와 논리적 데이터베이스 구조는 어플리케이션 유형과 목적에 부합되어야 한다.[5] 또한, 어플리케이션과 질의문 뿐 아니라 처리하는 데이터의 파악 역시 중요하다. 대부분의 경우, 어플리케이션 개발자는 어플리케이션 데이터의 특징에 대하여 더 많이 알고 있다. 그러므로, 어떤 상황에서 시스템 관리자는 옵티마이저가 결정한 실행계획을 분석 후, 그것보다 더 효율적인 방식을 선택하여 질의문의 재작성이나 변경을 함으로써 시스템의 성능을 향상 시킬 수 있다. 성능 향상을 위한 단계별 지침은 현상 조사 및 문제 분석, 문제 해결 방안 수립, 문제 해결 및 성능 향상의 순으로 전개 되어진다.

6. 결론

분류된 유형별 데이터의 저장 및 효과적인 처리를 위한 시스템 중 관계형 데이터베이스 시스템이 큰 비중을 차지하며 오랫동안 사용되어져 왔다. 이러한 시스템을 운영하는데 있어서 본문에서 논의 된 바와 같이 관계형 데이터베이스 시스템의 성능을 저하 시키는 요인은 여러 가지가 있었다. 이러한 요인을 고려하지 않은 시스템의 구축은 시스템 운영시 막대한 부하를 초래하고 기업의 경쟁력을 약화 시키는 주요 원인이 된다. 특히, 소개된 요인 중 응용 프로그램에 의하여 발생되는 시스템 성능 저하 요인은 60%를 차지 하고 있었다. 본 논문에서는 데이터베이스 튜닝을 위한 기본 지침과 특히, 어플리케이션 튜닝을 적용하여 시스템의 성능향상을 도모 하였고, 튜닝의 방법과 고려 사항들을 제시하였다. 이러한 종류의 튜닝을 통하여서 관계형 데이터베이스 시스템은 특정 데이터나 작업, 혹은 사용자의 적절한 목적에 맞추어서 최대한의 성능으로 구동 될 수 있을 것이다.

7. 참고문헌

[1] A First Course in Database Systems, Jeffrey D. Ullman and Jennifer Widom, Prentice-Hall, 1997.
[2] Oracle 8 : Database Administration, 성능튜닝 워크숍, SQL 튜닝, ORACLE, 1998.
[3] Database Tuning, Dennis E. Shasha, Prentice Hall, 1992.
[4] Tuning을 통한 데이터베이스 시스템 성능 향상, 최용락, 1996.
[5] http://dblab.changwon.ac.kr/oracle/tuning/, Oracle Server Tuning.