논문 2010-3-4

정보시스템 성능 향상을 위한 SQL 튜닝 기법

SQL Tuning Techniques to Improve the Performance of Integrated Information Systems

김양진*, 주복규**

Yang-Jin Kim, Bok-Gyu Joo

요 약 통합 정보시스템의 도입과 구축에 있어 가장 중요한 성공요소중의 하나가 성능 최적화이다. 본 연구에서는 정보시스템의 성능향상을 위해 비용도 저렴하고 단시간에 효과를 나타낼 수 있는 관계형 데이터베이스의 SQL 튜닝 기법을 제안하고, 이를 중소규모의 회사에 실제 운용되고 있는 데이터베이스 시스템에 적용하여 그 효율성을 분석하였다.

Abstract One of the most critical success factor in introducing and operating an integrated information system is the performance management. In this study, we propose some SQL tuning techniques, applicable to optimize the performance of relational database systems. We showed effectiveness of the techniques by applying to a database system of a medium scale company.

Key Words : Relational database, Oracle 10g, SQL tuning

Ⅰ. 서 론

데이터베이스란 용어는 1963년 6월에 미국 SDC (System Development Corporation)가 개최한 제 1차 '컴퓨터 중심의 데이터베이스 개발과 관리' 심포지움 제목에서 처음 사용되었다. 우리나라에는 1990년대 초반부터 활발히 도입되어 90년대 중반 이후부터 정보시스템의 근간으로 자리 잡았다[1]. 특히, 전사적 자원 관리(ERP), 고객 관계 관리(CRM), 데이터 웨어하우스(DW)와 같은 기업의 전략적 정보시스템이 국내에서도 활발히 도입되고 있는데, 이들 정보시스템의 핵심은 데이터베이스이다. 이와 같은 대규모 정보시스템의 도입과 구축에 있어 가장 중요한 성공 요인 중의 하나가 성능 관리이다. 아무리 훌륭한 비즈니스 개념이 정보시스템에 구현되더라도 데이터베이스에 존재하는 여러 종류의 대용량 데이터를 수많은 사람들이 동시에 접근하는 정보시스템의 성능관리에 실패한다면 결국은 실패한 정보시스템으로 판명날 것이기 때문이다.

정보 시스템의 초기 개발 및 운용 단계에서는 개발기간을 준수해야 하기 때문에 시스템의 속도에 대한 고려가 충분히 반영되지 않는다. 하지만 시스템을 운영하면서 데이터가 축적됨에 따라 시스템이 느려지게 된다. 따라서 대부분의 기업이 유지 보수 단계에서 성능을 평가하여 최적화하는 등 성능관리에 주력하게 된다.

회사 운영의 핵심 도구인 정보시스템의 성능개선을 위하여 기업들은 네트워크 대역폭 증설, 컴퓨터 하드웨어 증설, DBMS 업그레이드, 시스템 재개발로 처리속도의 위기를 해결하고 있다. 그러나 이러한 방법은 많은 비용을 수반할 뿐만 아니라 데이터가 지속적으로 축적됨에 따라 또다시 처리속도가 저하되는 등 근본적인 해결책이 되지 못하고 있는 실정이다[2, 3].

*준회원, 홍익대학교 컴퓨터정보통신공학과
**종신회원, 홍익대학교 컴퓨터정보통신공학과 교수
접수일자 2010.04.17 수정일자 2010.06.01

데이터베이스 시스템의 개발자들은 대부분 개발 초기 단계에서 부족한 샘플 데이터를 가지고 시스템을 테스트를 한다. 이때는 비효율적인 SQL이더라도 원하는 결과를 순식간에 처리하므로 속도의 문제를 느끼지 못한다. 관계형 데이터베이스 시스템의 성능을 저하시키는 요인 중 60% 이상이 응용프로그램에서 적절하지 못한 SQL문의 사용에서 발생한다.

본 논문에서는 저렴한 비용과 단기간에 가시적인 개선 효과를 나타낼 수 있는 최적의 SQL 사용에 관한 튜닝 방법에 대해 연구하였다. 우리는 기 제안된 SQL 튜닝 방법들을 소개하고 이를 현재 사용되는 통합정보시스템에 적용하여 기존 시스템의 성능과 비교함으로써 시스템 성능개선에 효과적임을 보였다.

이 논문의 구성은 다음과 같다. 제 2장에서는 지금까지 제안된 데이터베이스 튜닝 기법들을 알아보고 이 기법들을 적용하여 성능 개선을 이룬 대표적인 결과를 소개한다. 제 3장에서는 구체적인 SQL 튜닝 기법을 중간 규모의 회사에 실제로 운용되고 있는 정보시스템에 적용해 보고 그 결과를 분석하였다. 4장은 결론과 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

데이터베이스에서의 튜닝이란 원하는 정보를 빠르게 출력하기위해 시스템을 조정하는 작업을 의미한다. 즉 데이터베이스 어플리케이션, 데이터베이스 자체, 운영체제 등의 조정을 통하여 기존보다 빠른 시간에 원하는 결과를 가져오도록 시스템 자원을 조율하여 데이터베이스 시스템의 성능을 향상 시키는 작업이다. 데이터베이스 튜닝은 갈수록 복잡화, 대량화되고 있는 시스템을 하드웨어 등 다른 요소는 그대로 유지 운영하면서 데이터베이스 시스템의 성능을 최적화는 방안으로 투자한 비용에 비해서 탁월한 효과를 거둘 수 있다는 점에서 크게 주목받고 있다.

튜닝은 현행 정보 시스템에 존재하는 문제점을 분석하고, 분석된 문제점을 단계적으로 해결함으로써 자원 활용을 극대화하여 정보시스템을 안정시킴으로써 사용자 만족과 관리능력을 향상시킨다[4]. 이장에서 우리는 지금까지 제안된 데이터베이스 튜닝 방법론을 알아보고, 실제로 웹기반 시스템과 통합정보 시스템을 대상으로 적용한 대표적인 사례를 소개한다.

1. 데이터베이스 튜닝 방법론

데이터베이스 튜닝을 위해서는 먼저, 튜닝 방법론에 따라 기존 시스템에 존재하는 문제점을 정확히 분석하고, 그 분석을 통해 튜닝의 목적과 대상을 정하고, 마지막으로 각 단계 별로 하나씩 해결해 나가야 한다. 또한, 실제와 유사한 환경에서 최소의 반복 테스트를 수행함으로써 튜닝 이전보다 나쁜 결과를 가져오지 않도록 해야 한다[5]. 데이터베이스 튜닝 방법론은 아래와 같이 크게 세 가지로 분류할 수 있다.

가. 비율 기반 분석

비율 기반 분석 방법론은 주로 1990년대 초반에 데이터베이스 컨설턴트들이 사용 했던 방법론으로 소규모 용량의 데이터베이스에서는 효율적인 방법이었다. 예를 들면, 데이터베이스 성능을 분석하기 위해서 시스템 관리자(DBA)가 주로 사용하는 방법은 데이터베이스 버퍼, 캐시 히트 비율로 이 값이 90% 이상이면 데이터베이스 성능에 문제가 없다는 식의 접근 방식이다.

그러나 최근 낮은 하드웨어 비용으로 인해 많은 데이터베이스 시스템의 데이터베이스 버퍼 캐시 히트 율이 99%가 넘는 경우가 많이 있지만, 이러한 경우에도 데이터베이스 성능을 저하시키는 병목 현상이 존재한다. 따라서 이 방법은 복잡한 데이터베이스 시스템 환경에서 데이터베이스 성능을 분석하는 데 한계가 있다.

나. 대기 이벤트 기반 분석

주로 1990년대 중반부터 사용된 대기 이벤트 기반 분석 방법론은 비율 기반 분석 방법보다는 좀 더 발전된 형태로써 데이터베이스에 연결된 동시 세션들이 데이터베이스의 어떤 자원에 대해 대기하고 있는가를 진단하고 이러한 대기 시간을 줄이는 접근 방법이다.

그러나 응답 시간에서 대기 시간만을 고려하고 서비스 처리 시간과 운영체제 자원에 대한 대기 시간은 고려되지 않으므로 서비스 처리 시간의 비중이 큰 데이터베이스 시스템을 분석하는 데는 한계가 있다.

다. 응답 시간 분석

응답 시간 분석 방법론에서 응답 시간은 처리 시간과 대기 시간으로 나눈다. 처리 시간에 대한 비중이 큰 데이

터베이스 시스템인 경우 대기 시간보다는 처리 시간을 줄이는 것이 낫다. 반대로 대기 시간에 대한 비중이 큰 데이터베이스 시스템인 경우에는 처리 시간보다는 대기 시간을 줄이는 것이 전체 시스템의 응답 시간을 개선하는 데 좋다.

응답 시간 분석 방법론은 2000년대부터 사용하기 시작한 분석 방법으로써 현재까지는 가장 효율적인 방법이다. 이 방법론에서 가장 중요한 요소는 분석 대상 시스템의 응답 시간 추이 분석이다. 어느 순간에 시스템이 최대한 사용 됐는지를 파악한 후 시스템, 데이터베이스, 프로그램, SQL 단위로 처리 시간과 대기 시간의 비중을 분석한다. 그림 1은 응답시간과 처리량과의 관계를 보여준다[5].

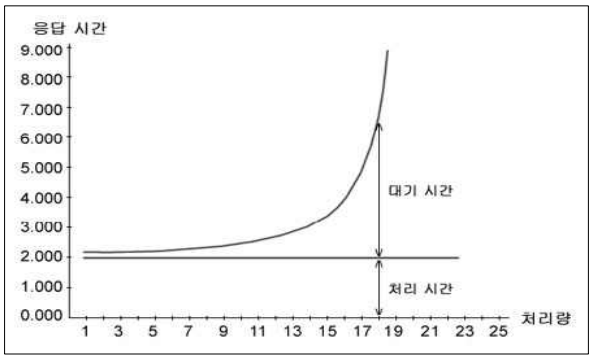

그림 1. 응답시간과 처리량의 관계
Fig. 1. Relationship between response time and throughput

2. 웹 기반 시스템의 튜닝

웹 기반 시스템에 튜닝 기법을 적용한 대표적인 사례는 다음과 같다. 연구 대상 시스템은 SQL Server 2000 E-Edition DBMS를 사용하고 Visual Studio 2003로 개발 운영 중인 웹 방식의 영업지원 시스템이다. 이 시스템의 사용자는 약 4만 명이며, 웹 서버 35대, DB 서버 9대, 버전관리 서버 1대, 코어 서버 2대로 운영 중인 시스템이다[6].

표 1. 웹 기반 시스템의 운영 환경
Table. 1. Web-Based System Environments

구 분	세부내용
DBMS	SQL Server 2000 E-Edition
SERVER	ES 570
Operating System	Windows 2003
User Interface	SQL Plus
DB Size	1.7TB
Tuning System	통합정보 시스템

이 시스템을 위해 제안된 SQL 튜닝 기법으로는 'sp_executesql' 활용, 'hint' 활용, 테이블 변수 활용, 통계 업데이트 자동 배치 작업 등이다. 이 튜닝 기법을 수행한 결과, 튜닝 전과 튜닝 후의 성능 평가에 대한 결과는 표 2와 같다.

표 2. 웹 기반 시스템의 튜닝 결과
Table 2. Tuning Results of Web-Based System

튜닝 방법	개선 분야	튜닝전	튜닝후
sp_executesql 활용	CPU 점유율 실행계획 생성	60% = SQL문 실행 횟수	35% 1회
hint활용	처리시간 I/O 비용 CPU 비용	546ms 3.19 0.393	123ms 0.00543 0.000179
테이블 변수 활용	CPU 점유율 처리 시간	65% 238ms	55% 156ms
통계 업데이트 자동배치 작업		수작업 통계 정보 생성	통계정보 자동생성 배치작업 실패율 감소

성능 평가 결과에 의하면, 튜닝 기법 수행 후 성능이 크게 향상된 것을 알 수 있다. 'sp_executesql'을 활용한 결과, 프로세서 점유율이 최대 58% 개선되는 것을 확인할 수 있었다. 'hint'를 활용한 결과, 23%의 처리 시간이 단축되는 것을 확인하였고, 테이블 변수를 사용한 결과, 프로세서 점유율이 최대 16% 개선되는 것을 확인 할 수 있었다. 그리고 통계 업데이트 자동 배치 작업 생성으로 인해 배치 작업 실패율이 감소하는 효과가 나타났다

3. 통합 정보 시스템의 튜닝

적용 연구 대상 시스템은 400명 정도의 직원의 회사로서, Oracle DBMS에 Windows NT4.5에서 운영 중인 시스템이다. Windows운영체제 PC를 단말기로 사용하며, Developer 2000 이나 SQL Plus를 사용자 클라이언트용으로 사용한다[7]. (표 3 참조)

표 3. 통합 정보시스템 운영환경
Table 3. Integrated Information System Environments

구 분	세부내용
DBMS	Oracle 7.3.4.5.0
SERVER	COMPAQ PROLIANT ML570
Operating System	Windows NT 4.5
User Interface	Developer 2000/ SQL Plus
DB Size	5.95GB
Tuning System	통합정보 시스템

이 시스템에 대하여 엑세스 튜닝, Logic튜닝 기법을 수행한 결과, 튜닝 전과 튜닝 후의 성능 평가에 대한 결과는 표 4와 같다.

표 4. 통합 정보 시스템의 튜닝 결과
Table 2. Tuning Results of Information System

튜닝 방법	개선 분야	튜닝전	튜닝후
뷰를 통한 조인 감소	데이터 처리량 처리 시간	15,966 10.07	878 2.47
hint활용	데이터 처리량 처리 시간	33,203 0.09	5,267 0.05
Logic 수정	처리 시간 (사용 sql문)	51.12 (Cursor)	29,32 (Sub Query)
Logic 수정	처리 시간 (사용 sql 문)	3.07 (for loop)	0.02 (outer join)

같은 작업을 처리하는데 데이터 처리량의 감소는 처리 시간의 감소를 의미한다. 결국 처리 시간을 줄인다는 것은 처리해야할 데이터양을 줄이는 것과 동일하다. 또한 절차적으로 구성되어 여러 번의 SQL을 호출하는 어플리케이션을 하나 또는 최소한의 SQL을 호출하는 집합적 어플리케이션으로 변환시킴으로써 처리속도를 단축시킬 수 있다.

Ⅲ. SQL 튜닝 기법의 적용

이 연구를 위하여 우리는 수도권의 한 중소업체의 통합정보시스템을 연구대상으로 선정하였다. 운영 중에 발생하는 성능 저하 원인을 토대로 이미 발표된 튜닝기법과 제안된 SQL 튜닝기법의 적용으로 얻어진 성능개선 효과를 비교하는 방식으로 연구를 진행하였다[8, 9].

1. 시스템 운영 환경

대상 시스템은 표 5에서와 같이 Oracle DB 10g를 사용하고 Windows 2000 Server에서 운영된다. 중소 규모 회사의 통합정보 시스템으로 인사, 급여, 근태, 자재관리, 재고관리, 고객관리, 코드관리로 구성되어 있고 각각의 데이터를 관리하는 테이블 30개로 구성되어 있다.

표 5. 시스템 운영 환경
Table 5. System Operating Environments

구 분	세부내용
DBMS	Oracle 10g
SERVER	IBM X86
Operating System	Windows 2000 SERVER
User Interface	SQL Plus
Tuning System	통합정보 시스템

2. SQL 튜닝 기법의 적용

우리는 대상시스템에 대표적인 SQL 튜닝의 상세한 기법들을 적용하였고 그 결과를 요약 기술한다. 더 상세한 연구 결과는 연구보고서에 있다[10].

가. WHERE 절 왼쪽 칼럼에 함수 사용

튜닝에 있어서 가장 기본이 되는 기법의 하나로 SQL 문의 WHERE 절에서 왼쪽 칼럼에 함수를 사용하게 되면, 기본적으로 인덱스를 사용하지 않는다. 따라서 데이터가 많을수록 처음부터 WHERE 절의 내용을 비교해야 하기 때문에, 아래 부분에 위치한 데이터를 처리할 경우에는 처리시간이 더 소요된다.

```
SQL> SELECt COUNT(*)
  2  FROM LKDM_Z50
  3  WHERE SUBSTR(ZIP_1, 1, 2) = '서울';

  COUNT(*)
----------
      2151

경   과: 00:00:00.23
```

그림 2. WHERE 절의 경우(튜닝 전)
Fig. 2. Case of WHERE Clause(Before Tuning)

```
SQL> SELECT COUNT(*)
  2  FROM LKDM_Z50
  3  WHERE ZIP_1 LIKE '서울%';

  COUNT(*)
----------
      2151

경   과: 00:00:00.01
```

그림 3. WHERE 절의 경우(튜닝 후)
Fig. 3. Case of WHERE Clause(After Tuning)

그림 2와 그림 3은 내용이 동일한 두 가지 SELECT 문으로, 함수를 사용한 경우와 사용하지 않은 두 경우를 보여준다. 카운트가 서로 2151건으로 같지만, 처리시간이 함수를 사용하지 않고 인덱스를 사용한 SELECT 문이 빠름을 알 수 있다. 또한 데이터가 많을수록 처리속도 차이는 더 벌어진다.

SQL문을 작성하다 보면, WHERE 절 왼쪽 칼럼에 함수를 써야할 경우가 많다. 처리속도에 큰 영향이 없다면 얼마든지 사용할 수 있겠지만 데이터량이 많아져서 속도 차이가 많이 난다면 다른 방법을 사용하는 것이 튜닝의 기본이다. 이 외에도 칼럼을 붙여서 사용하거나 계산 등을 하는 것도 튜닝 원칙에 위배 된다.

나. IN 과 EXISTS

IN 과 EXISTS는 비슷한 용도로 자주 사용된다. 튜닝 상으로는 EXISTS 문이 더 효과가 있다. 특히 IN의 경우에 IN 다음의 SELECT 문의 결과 내용이 많을수록 조인(join) 형태인 EXISTS 문이 적당하다. 그림 4와 그림 5는 EXISTS 문의 효과를 보여준다.

```
SQL> UPDATE LKDM_D01
  2  SET BIGO = 'SS'
  3  WHERE PART_CODE IN
  4  (SELECT PART_CODE
  5  FROM LKDM_E01
  6            WHERE      INOUT_DATE      BETWEEN
TO_DATE('19990101','YYYYMMDD')
  7  AND TO_DATE('19991231','YYYYMMDD'));

24 행이 갱신되었습니다.

경   과: 00:00:00.12
```

그림 4. IN을 사용한 경우(튜닝 전)
Fig. 4. Case of Using IN(Before Tuning)

```
SQL> UPDATE LKDM_D01 A
  2  SET A.BIGO = 'SS'
  3  WHERE EXISTS(SELECT PART_CODE
  4  FORM LKDM_E01
  5  WHERE PART_CODE = A.PART_CODE
  6  AND INOUT_DATE BETWEEN
       TO_DATE('19990101','YYYYMMDD')
  7  AND TO_DATE('19991231','YYYYMMDD'));

24 행이 갱신되었습니다.
경   과: 00:00:00.07
```

그림 5. EXISTS을 사용한 경우(튜닝 후)
Fig. 5. Case of Using EXISTS(After Tuning)

다. 부등호 사용

SQL 문에서 부등호의 사용은 함수의 사용과 마찬가지로 처리량 증가의 원인이기 때문에 조건 절에서 범위를 주는 경우 BETWEEN 문을 사용한다. 그리고 '아니다'라는 뜻의 '<>' 부호는 UNION과 EXISTS를 이용해서 작성할 수도 있다. 그림 6과 그림 7은 부등호 대신 BETWEEN을 사용한 효과를 보여준다.

```
SQL> SELECT COUNT(*)
  2  FROM LKDM_Z50
  3  WHERE SUBSTR(ZIP_1, 1, 1) >= '경'
  4  AND SUBSTR(ZIP_1, 1, 1) <= '고';
  COUNT(*)
----------
      2285
경   과: 00:00:00.03
```

그림 6. 부등호를 사용한 경우(튜닝 전)
Fig. 6. Case of Using Inequality(Before Tuning)

```
SQL> SELECT COUNT(*)
  2  FROM LKDM_Z50
  3  WHERE ZIP_1 >= '경'
  4  AND ZIP_1 <='고';
  COUNT(*)
----------
      2285
경   과: 00:00:00.01
SQL> SELECT COUNT(*)
  2  FROM LKDM_Z50
  3  WHERE ZIP_1 BETWEEN '경' AND '고';
  COUNT(*)
----------
      2285
경   과: 00:00:00.00
```

그림 7. BETWEEN을 사용한 경우(튜닝 후)
Fig. 7. Case of Using BETWEEN(After Tuning)

라. FROM 절 테이블 위치

일반적으로 SELECT문에서 많은 테이블을 조인해서 사용하는데, FROM 절에서 테이블의 위치를 기본이 되는 테이블 순으로 위치시킨다. 이것은 일관성을 주어 깔끔한 정렬을 하기 위함이나, 튜닝 상으로는 반대로 하는 것이 더 좋다.

FROM절에 나열된 테이블이나 인라인 뷰의 순서가 기준이 되는 테이블이 나중에 올수록 처리속도가 빨라진다. 모든 경우가 그런 것이 아니지만, 대부분의 경우에 처리속도가 향상된다.

표 6. 실험 결과 요약
Table 6. Summary of Tuning Applications

튜닝방법	튜닝전	튜닝후
Where 절 함수사용	00:00:00.23	00:00:00.01
부등호사용	00:00:00.14	00:00:00.01
In과 Exists	00:00:00.12	00:00:00.07
From절 테이블정렬	00:00:00.16	00:00:00.07
From절 테이블정렬	00:00:01.50	00:00:00.25

3. 적용 결과의 평가

튜닝은 어떠한 규칙이 있더라도, 연관된 테이블 상황, 네트워크의 상태, 컴퓨터 사양 등으로 항상 최적화 된 다고 볼 수는 없다. 따라서 상황마다 다양한 테스트를 통해서 정확히 분석해야 한다.

이번 실험에서는 데이터베이스의 데이터 량이 많지는 않아서 큰 차이가 나오지는 않았지만, 표 6에서 보는바와 같이 기본적인 튜닝 기법의 적용에도 눈에 띄게 처리 속도의 향상이 있음을 알 수 있다. 시스템을 변경 하지 않고 함수의 사용 변경이나 테이블의 재배치 등으로 처리속도의 향상을 가져왔음으로 데이터 량이 많아지면 더 큰 효과를 기대 할 수 있다.

Ⅳ. 결 론

기업의 경영활동에 있어 정보의 수집, 축적, 가공 및 효과적인 활용은 필수적이다. 점차 대형화, 복잡화되어가는 기업의 정보시스템에서 관계형 데이터베이스는 데이터의 관리와 활용에 있어 기존의 파일시스템과 비교할 때 매우 간편하고 강력한 기능을 제공한다. 그러나, 데이터베이스의 성능은 활용하는 수준에 따라 구현된 성능의 편차가 심하게 나타나며, 대부분의 성능저하는 적절하지 못한 SQL 구문의 사용에 기인한다.

본 연구에서는 동일한 조건으로 가시적인 효과를 발생할 수 있는 구체적인 SQL 튜닝 기법을 중간 규모의 회사에 실제로 운용되고 있는 정보시스템에 적용해 보고 그 결과를 분석하였다. 이 연구에서 우리는 주요 SQL 튜닝 기법이 성능개선 효과가 있음을 확인하였다. 그 구체적인 기법이란 SQL 문에서 칼럼에 함수사용, 부등호 사용, IN과 EXISTS, 테이블 재배치, 인라인 뷰를 적용하기 등이다. 그 결과 실험 대상 시스템에서 적은 시간과 노력의 투입에도 불구하고 시스템의 오버헤드가 줄어들었고, 서버의 자원 활용률이 크게 개선되어 사용자의 접속처리가 향상 되었다.

향후에는 Oracle을 중심으로 진행된 본 연구를 또 다른 상용 관계형 데이터베이스 제품인 MS SQL Server등에서도 활용이 가능한 일반화된 기법으로의 전환이 필요하며, 튜닝을 위한 사전 진단 및 활용 절차에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] 이석호, 데이터 베이스 론, 정익사, 2006.
[2] 조종암, 대용량 데이터베이스를 위한 오라클 SQL 튜닝, 대청, 2000.
[3] 최송희, SQL튜닝을 이용한 데이터베이스 시스템 성능개선에 관한 연구, 경희대학교 산업정보대학원, 2003.
[4] 정미영, SQL튜닝을 이용한 처리속도 향상 요소에 대한 고찰 및 성능 측정, 홍익대학교 정보대학원, 2007.
[5] DBguide.net, www.dbguide.net.
[6] 안지현, 웹 기반 시스템 성능 향상을 위한 관계형 데이터베이스 SQL 튜닝 연구, 성균관대학교 정보통신대학원, 2007.
[7] 임혁민, SQL 튜닝을 이용한 처리속도 향상 관한 실증연구, 동국대학교 정보산업대학원, 2001.
[8] 정재준, 오라클 실무활용 SQL 튜닝, 혜지원, 2008.
[9] 이공명, 오라클 프로젝트 실무, 컴스페이스, 2003.

[10] 김양진, 통합정보 시스템의 성능향상을 위한 관계형 데이터베이스의 SQL 튜닝 기법에 관한 연구, 홍익대학교 컴퓨터정보통신공학과, 2009년 12월.

※ 이 논문은 2009학년도 홍익대학교 학술연구조성비에 의하여 연구되었음

저자 소개

김 양 진(준회원)

- 2010년 2월 홍익대학교 컴퓨터정보통신공학과(학사)

<주관심분야 : 데이타베이스>

주 복 규(종신회원)

- 1977년 서울대학교 계산통계학과 (학사)
- 1980년 한국과학원 전산학과(석사)
- 1990년 메릴랜드대학교 전산학과(박사)
- 1990년~ 1998년 삼성전자 중앙연구소 수석연구원
- 1998년~ 2000년 (주)동양시스템즈 연구소장
- 2001년~ 현재 홍익대학교 컴퓨터정보통신공학과 교수

<주관심분야 : 소프트웨어 재사용, 인터넷 보안>