The Journal of Korean
Institute of CALS/EC
Vol 7, No. 1, April 2002

데이터베이스 시스템 성능 향상을 위한 데이터베이스 관리 시스템 파라미터 튜닝 프로세스

최용락*, 윤병권*, 정기원*

Database Management System Parameter Tuning Processes for Improving Database System Performance

Yong-Lak Choi, Byungkwon Yoon, Kiwon Chong

Abstract

Database system parameter tuning is one of database system tuning that achieve to improve performance of database system with application program tuning and data model tuning. By parameter tuning adjusts value of entry that is stored in data dictionary's parameter file that is included to database system, it is thing which make relevant database system can display performance of most suitable. And, it is that achievement is one of possible tuning method immediately without occurrence of additional expense or involved hardware for database system performance elevation and ashes composition of software. But, it is actuality that administration about parameter practical use is not achieved, and is using Default Value of parameter that database management system offers just as it is systematically. So, this paper presents parameter tuning process that can achieve parameter tuning of database system that is operating present systematically, and parameter tuning process each activity important input urea and tuning achievement product. And explain about effect and result that happen by sort database system performance and parameters that it is affinity systematically, and grasp relationships between parameter, and change parameter of string database system. And not that parameter uses contents that specify by fixing when establish database administration system, is going to emphasize and explain that must utilize changing continuously during database system operation. It changes parameter entry value how in various kinds different operation environment and present if must apply, and will arrange effect that this parameter entry value alteration gets in performance taking into account point that is actuality that is using parameter that define database administrators when install the database system just as it is continually without alteration.

* 숭실대학교 대학원 컴퓨터학과

1. 서론

기업 또는 조직간에 경쟁이 더욱 심화되고 기업 환경도 빠르게 변화되는 정보산업 사회에서 정확하고 신속한 정보의 제공은 기업의 경쟁력을 강화하는 필수적인 사항으로 인식되고 있으며, 기업에서 필요한 정보를 체계적으로 관리하고 사용성을 높이기 위해 정보의 수집 및 처리 단위가 기존의 해당업무에 대한 단순 데이터 질의 처리에서 전사적인 통합/대형화 추세에 있다[3].

이에 따라 대부분의 기업 또는 조직은 데이터베이스 시스템의 활용을 필수 요소로 인식하고 있으며, 현재 운용 중이거나 개발할 데이터베이스 시스템의 성능이 전체 정보시스템의 성능에 많은 영향을 미치기 때문에 데이터베이스 시스템의 성능을 향상시키는 일에 많은 노력과 비용을 투자하고 있다. 그러나 데이터베이스의 성능 향상을 위한 관련 연구는 응용 프로그램, 데이터 모델 등 부분적으로는 많은 진전이 있으나 데이터베이스 시스템의 파라미터는 데이터베이스 관리 시스템(DBMS: DataBase Management System)에 종속적이기 때문에 이에 대한 연구가 부족하며 대부분의 경우 데이터베이스 관리 시스템이 제공하는 파라미터의 기본 값(Default Value)을 그대로 사용하고 있는 것이 현실이다. 그러나 파라미터 값을 설정하는 작업은 응용 프로그램이나 데이터 모델에 비해 적은 비용과 노력으로 수행할 수 있으므로 간과되어서는 안될 부분이다. 이에 따라 본 논문에서는 데이터베이스 튜닝 대상 중에서 데이터베이스 시스템 관리를 위한 데이터베이스 시스템 파라미터 튜닝을 방법론적 절차에 따라 문제점을 분석하고, 분석된 문제점을 단계적으로 해결함으로써 데이터베이스 관련 자원 활용을 극대화할 수 있는 파라미터 튜닝 단계의 프로세스를 제시한다.

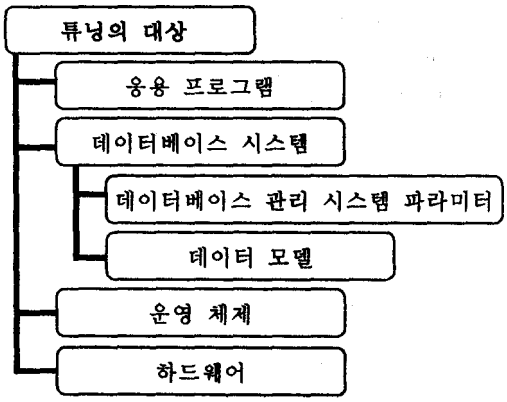
본 논문의 구성은 2장에서 데이터베이스 튜닝의 일반적인 관련 연구 사항을 다루고, 3장에서는 체계적인 데이터베이스 관리 시스템의 파라미터 튜닝을 위한 프로세스를 제시하며 각 프로세스의 수행 활동을 제시하고 각 활동에 대한 작업(태스크:Task)을 설명 하고 있다. 마지막으로 4장에서 결론을 기술 하였다. 그리고 본 논문에서는 오라클 데이터베이스 관리 시스템의 파라미터를 기준으로 예제를 제시하였다.

2. 관련 연구

2.1 튜닝(Tuning)의 개요

정보 시스템 관리자는 응용 프로그램, 데이터베이스 시스템, 그리고 운영 체제의 조율 등을 통하여 정보 시스템의 성능을 향상 시킬 수 있다. 이러한 조율 과정을 튜닝이라고 하며 적절한 튜닝 수행을 통하여 최적의 정보 시스템 성능을 유지할 수 있는 것이다. 데이터베이스 시스템 튜닝도 정보 시스템의 성능을 향상 시키기 위하여 수행하는 튜닝의 한 종류이며 <그림 1>은 데이터베이스 시스템 관련 튜닝의 대상을 나타낸 것이다[2].

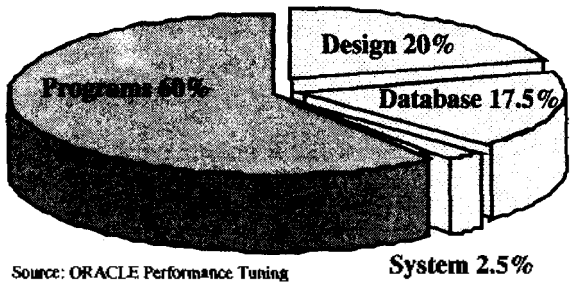
먼저 응용 프로그램 튜닝은 실제로 구현되어 활용되는 모든 일반 사용자 프로그램과 내장형 프로그램 등을 포함한다. 다음으로



<그림 1> 데이터베이스 시스템 관련 튜닝의 대상[1]

데이터베이스 시스템 튜닝은 데이터베이스 관리 시스템 파라미터 튜닝과 데이터 모델 튜닝으로 나뉘어지며, 데이터베이스 관리 시스템 파라미터 튜닝은 데이터베이스 시스템 운영에 필요한 모든 데이터베이스 관련 파라미터를 조율하는 것으로써 하드웨어 시스템 관련 파라미터를 포함하여 사용자 관리 관련 파라미터 등의 모든 자원(Resource)에 대한 관리를 수행하는 파라미터 값(Value)을 조정하는 것이다. 그리고 데이터 모델 튜닝은 데이터베이스의 논리적/물리적 스키마를 조정하여 데이터베이스 시스템의 성능을 향상 시킬 수 있는 방법을 모색하고 현재 사용중인 데이터베이스 시스템에 새로운 사용자 요구나 운영 방법의 변화 또는 정보 시스템에 영향을 주는 환경의 변화 등을 반영하여 새로운 데이터베이스 시스템 구조를 도출하는 것이다. 운영 체제 튜닝은 현재 사용하고 있는 운영 체제의 종류 별로 많은 차이를 보이며 시스템 성능에 영향을 미치는 요인들이 서로 다르다. 따라서 운영 체제 관련 전문가와 같이 수행해야 한다. 끝으로 하드웨어 튜닝은 파라미터 튜닝과 밀접한 연관 관계가 있는 튜닝으로써 하드웨어의 전체적인 구성 상태와 프로세서, 메모리, 디스크, 입/출력 컨트롤러 등에 대한 활용률, 활용 방법 등을 고려하여 진행해야 하는 단계이다. 아울러 하드웨어 튜닝 단계에서는 네트워크 관련 자원 및 운영 상태에 대한 튜닝도 수행해야 한다.

<그림 2>는 관계형 데이터베이스 시스템의 성능 저하에 영향을 미치는 요인들을 나타낸다.



<그림 2> 관계형 데이터베이스 시스템의 성능 저하에 영향을 미치는 요인[6]

그림에는 응용 프로그램이 60%라는 범위로 가장 큰 비중을 차지하고, 데이터베이스 설계가 20%의 비율, 그리고 데이터베이스 관리 시스템 부분이 17.5%를 차지하는 것으로 표현되었다[5]. 특히 데이터베이스 관리 시스템 부분은 데이터베이스 시스템의 관리 및 운영에 필요한 데이터베이스 관련 파라미터를 포함한 데이터베이스 관리 시스템의 요인 들을 의미하는 것이다. 그리고 하드웨어 부분이 2.5% 정도로 표현되어있는데 이 부분은 데이터베이스 튜닝 과정 및

단계에서 최후에 고려해야 하는 사항이다. 응용 프로그램 부분이 데이터베이스 시스템 성능에 가장 큰 영향을 미치는 것으로 나타나 있으며 대부분을 차지하는 것으로 표현되어있으나 각 요소들은 상호 연관 관계가 있으며 상호 보완적인 관계에 있으므로 각 요소들 모두가 중요하다는 것을 인식해야 한다. 예를 들어 데이터베이스 설계를 바꾼다면 관련 응용 프로그램은 당연히 바뀌게 될 것이고 새로운 데이터베이스 구조에 적합하게 작성되어야 할 것이다. 아울러 데이터베이스 시스템을 관리하는 관련 파라미터가 변경된다면 응용 프로그램에서 데이터를 찾는 방법이나, 처리하는 방법이 달라지게 될 것이다. 또한 데이터베이스 설계를 조율하면 응용 프로그램 튜닝 단계에서 고려해야 할 시스템 성능 저하 요인들을 미리 제거하고 수정할 수 있으므로 보다 효율적인 튜닝을 진행할 수도 있게 될 것이다[8].

2.2 파라미터

2.2.1 개요

인스턴스는 백 그라운드 프로세스(Back Ground Process)와 데이터베이스 운용에 활용하는 메모리 구조를 합쳐서 말하는데 모든 인스턴스는 데이터베이스가 실행되기 위해서 파라미터 파일을 참조한다. 이 파라미터 파일은 인스턴스가 시작될 때 SQL* PLUS와 같은 툴에 의해서 이용된다. 파라미터 파일은 파라미터 앤트리 리스트와 그 파라미터에 해당하는 값으로 이루어져 있다. 예를 들어, PROCESSES = 100 이러한 형태의 리스트로 되어있다. 임의의 파라미터 값을 수정하게 되면 데이터베이스 서버를 정지 했다가 다시 시작해야 한다. 하지만 일부의 동적 파라미터는 해당 세션에서 수정되어 사용될 수 있다. 오라클에 사용되는 파라미터 값은 모두 선택적으로 사용되고 오라클 서버는 이미 각 파라미터에 대한 기본값을 가지고 있으며, 파라미터 파일의 파라미터들은 순서에 관계없이 기술 될 수 있고 중복해서 파라미터가 쓰여질 경우 앞에 쓰여진 파라미터의 값은 뒤에 쓰여진 파라미터에 의해 값이 덮어 씌워진다. 또한 IFILE 파라미터를 이용해서 다른 파라미터 파일을 읽어 들일 수 있다. 다음은 ORACLE 데이터베이스 관리 시스템을 기준으로 한 백 그라운드 프로세스의 예이며 백 그라운드 프로세스 들이 활용하는 메모리 공간을 SGA(System Global Area)라 하며, SGA를 관리하는 파라미터들이 데이터베이스 시스템 성능에 가장 많은 영향들을 미치는 파라미터들이다[7].

① SMON(System MONitor) : 데이터베이스 가동 시에 복구 작업을 수행하는 프로세스이며, 여러 개의 데이터베이스가 같은 데이터 파일을 공유할 수도 있으며 다른 컴퓨터의 손상된 데이터도 복구 가능하다. 데이터베이스가 오픈 되었을 때 일관성을 검사하고 데이터베이스 복구를 시작하는 것이 주 기능이며 자동 인스턴트 복구, Temporary Segment에 의해 이미 사용되었거나 더 이상 사용되지 않는 공간을 재할당 작업수행을 하고 연속 공간으로 합병하는 작업(Coalescing)을 수행하는 필수적 프로세스 이다.

② PMON(Process MONitor) : 사용자 세

션이 비정상적으로 종료(수행하던 작업 실패)되었을 때 사용하던 데이터베이스에 대한 복구 작업을 수행하는 프로세스로써 사용자에게 할당되었던 자원을 반납하고 트랜잭션을 롤백을 수행하는 필수적 프로세스이다.

③ DBWR(DataBase WRiter) : 변경된 데이터 블록을 데이터베이스 파일에 옮겨 기록하기 위한 필수적 프로세스 이다.

④ LGWR(LoG WRiter) : 리두(Redo) 내용을 리두 로그에 기록하며 리두 로그 버퍼에 등록된 변경 사항을 읽어서 데이터베이스(리두 로그 파일)에 기록하는 필수적 프로세스 이다.

⑤ CKPT(CheckK PoinT) : 사용자가 데이터를 수정하거나 결과를 원래의 데이터 파일에 재 기록하고자(영구적으로 기록) 할 때 메모리 영역의 수정된 데이터 블록(Dirty Data Block)이 디스크에 기록되도록 하며 컨트롤 파일에 리두 로그의 전환 내용을 전달하며 필수적 프로세스 이다.

2.2.2 파라미터의 이용 및 형태

데이터베이스 관리 시스템은 파라미터를 아래와 같은 목적으로 사용한다[9].

- 전체 데이터베이스의 제한 설정
- 사용자 또는 프로세스 제한 설정
- 데이터베이스 시스템에서 요구되는 파일명과 디렉토리 설정
- 데이터베이스의 컨트롤 파일 설정
- 국가 언어 지원 내용 설정
- 메모리 구조를 위해 저장장소 할당
- 동적 파라미터를 이용한 데이터베이스 성능 개선

데이터베이스 관리 시스템에서 사용하는 파라미터의 형태는 다음과 같다.

- Derived Parameters:
  다른 파라미터의 값으로부터 계산되어 값이 바뀌는 파라미터
- Global Cache Parameters with the Prefix GC:
  접두어 GC가 붙은 파라미터로 Oracle Parallel Server 시스템에 사용되는 파라미터
- Operating System Dependent Parameters:
  몇몇 파라미터는 운영체제에 의존하여 값의 범위와 유효한 값 결정
- Variable Parameters or Dynamic Parameters:
  시스템 성능을 향상 시키기 위해 가장 잠재성을 지닌 파라미터
- Heterogeneous Services Parameters:
  Oracle Heterogeneous Services를 위한 파라미터

2.2.3 동적 파라미터 (Dynamic Parameter or Variable Parameter)

일부 파라미터는 유동적이기 때문에 동적 파라미터라 하고 동적 파라미터는 인스턴스가 실행되는 동안 ALTER SESSION, ALTER SYSTEM, ALTER SYSTEM DEFERRED 등의 명령을 이용하여 파라미터 값을 수정할 수 있다.

ALTER SESSION 명령으로 변경한 파라미터는 현재 세션이 진행되는 동안에만 유효하다. 즉, 현재 인스턴스에 있는 다른 세션에는 영향을 주지 않는다. ALTER SYSTEM

명령으로 변경한 파라미터는 현재 인스턴스에 있는 모든 세션에 영향을 준다. ALTER SYSTEM DEFERRED 명령으로 변경한 파라미터는 현재 인스턴스에 있는 세션에는 영향을 주지 않고 미래에 모든 세션에 영향을 미친다. 이와 같은 동적 파라미터의 변경은 업무 처리의 성능 개선을 위하여 운영되는 응용 프로그램의 정확한 이해를 통해 원하는 시점에 변경 사용하여야 한다[7],[8].

2.2.4 파라미터의 파악 및 분류

먼저 파라미터 튜닝을 위하여 해당 데이터베이스에서 활용하는 파라미터에 대한 기본적인 내용을 파악하여 자료화 해야 할 것이다. 물론 데이터베이스의 파라미터는 모든 데이터베이스 들이 서로 다르므로 파악된 내용이 데이터베이스 별로 서로 상이할 수는 있으나 궁극적으로 모든 파라미터 들이 서로 연관이 있으며 상호 연관 운용되어 데이터베이스 최적의 운영 환경을 제공하기 위한 것이라는 것을 알게 될 것이다. <표 1>은 파라미터의 기본 정보를 파악하여 정리한 예제이다.

<표 1> 오라클 데이터베이스 관리 시스템의 파라미터 기본 정보

DB_BLOCK_SIZE	SGA에 할당되는 Data Block의 크기 (Byte)	필수	O/S 종속	2048	32768	
				변경 불가		
DB_BLOCK_BUFFERS	SGA에 할당되는 Data Block의 개수	필수	60	4	O/S 종속	
		Row Cache Hit Ratio 0.95 이하		Parameter 조정	수시	
SHARED_POOL_SIZE	SGA의 공유 풀의 크기 (Byte)	필수	3.5M	300K	O/S 종속	OPEN_CURSORS
		Library Cache Hit Ratio가 0.95 이하		Parameter 조정		
BACKGROUND_DUMP_DEST	Background Process들의 Trace File이 저장될 위치	필수	O/S 종속			
		DB 생성시		변경 불가		
CONTROL_FILES	데이터베이스 관리용 Control File의 명칭 과 위치	필수	O/S 종속	1 File	8 Files	
		DB 생성시		변경 불가		
DB_NAME	데이터베이스 식별자로 사용될 데이터베이스 생성시 지정	필수	None			
		DB 생성시		변경 불가		
USER_DUMP_DEST	사용자의 Trace File이 저장될 위치	필수	O/S 종속			
		DB 생성시		변경 불가		
COMPATIBLE	Instance의 호환되어야만 하는 Server의 Version	필수	6.0.0	Default Release	Current Release	
		DB 생성시		변경 불가		
IFILE	Parameter File에 저장된 다른 Parameter File의 명칭 및 위치	필수	None			
		DB 생성시		변경 불가		
LOG_BUFFER	SGA의 REDO Log Buffer에 할당되는 크기 (Byte)	필수	[illegible]	O/S 종속	O/S 종속	[illegible]
		Log Buffer의 경합 발생시		Parameter 조정		
MAX_DUMP_FILE_SIZE	O/S의 Block 개수로 지정되는 Trace File의 최대 크기	선택	Unlimited	0	Unlimited	
		DB 생성시		변경 불가		
PROCESSES	Instance에 동시에 접속할 수 있는 O/S Process의 최대 개수	필수	PARALLEL_MAX_SERVERS	6	O/S 종속	SESSIONS TRANSACTIONS
		DB 생성시		변경 불가		
SQL_TRACE	모든 사용자 Session에 대한 SQL 추적 기능의 사용가능 여부	선택	False	False	True	
		필요 시		Parameter 조정		
TIME_STATISTICS	Trace File과 Monitor Screen에 시간정보 기록 기능의 사용 여부	선택	False	False	True	
		필요 시		Alter System		

파라미터는 여러 가지 방법으로 분류해 볼 수 있다. 먼저 파라미터가 활용되는 시기에 따라 분류해 볼 수도 있고, 또는 파라미터의 변경 가능 여부에 따라 가변/불변 분류, 그리고 데이터베이스 시스템 운영과 관련된 자원들에 영향을 미치는 부분에 따라서 분류해 볼 수 있다.

여기서는 각 파라미터가 메모리, 프로세스, 디스크, 네트워크, 사용자, 기타 서버 환경에 영향을 미치는 부분을 나타내고, 동적 파라미터로써 가변/불변 여부에 관하여 조사한 예제를 <표 2>에 나타내 보았다. 오라클 데이터베이스 관리 시스템의 경우에 약 180개 정도의 파라미터가 존재하며 이 모든 파라미터들이 관리 대상이 되어야 하지만 본 예제에서는 가장 중요한 파라미터 35개 정도의 예제를 작성했다. 그리고 예제에도 나타나 있듯이 각 파라미터는 반드시 한가지 서버 환경에만 영향을 미치는 것은 아니고, 하나의 파라미터의 값을 수정함으로써 여러 가지 서버 환경에 영향을 줄 수도 있다는 점을 이해해야 할 것이다.

<표 2> 파라미터의 분류와 동적 파라미터 여부

BACKUP_TAPE_IO_SLAVES				✓				0
BUFFER_POOL_KEEP		✓						X
BUFFER_POOL_RECYCLE		✓						X
CURSOR_SHARING		✓	✓					0
CURSOR_SHARING_FOR_TIME		✓	✓					X
DB_BLOCK_BUFFERS	✓	✓						X
DB_BLOCK_LRU_LATCHES	✓	✓	✓	✓				X
DB_BLOCK_MAX_DIRTY_TARGET				✓				0
DB_BLOCK_SIZE				✓				X
DB_FILE_DIRECT_IO_COUNT				✓				0
DB_FILE_MULTIBLOCK_READ_COUNT				✓				0
DB_WRITER_PROCESSES	✓			✓				X
DBWR_IO_SLAVES				✓				X
FAST_START_IO_TARGET				✓				0
LARGE_POOL_SIZE		✓		✓				X
LOG_BUFFER		✓						X
LOG_CHECKPOINT_INTERVAL				✓			✓	0
LOG_CHECKPOINT_TIMEOUT				✓			✓	0
MTS_CIRCUITS		✓			✓			X
MTS_DISPATCHERS			✓		✓			0
MTS_MAX_DISPATCHERS			✓		✓			X
MTS_MAX_SERVERS			✓		✓			X
MTS_SERVERS			✓		✓			0
MTS_SESSIONS					✓			X
OPEN_CURSORS		✓	✓					X
PRE_PAGE_SGA		✓						X
ROLLBACK_SEGMENTS	✓							X
SESSION_CASHED_CURSORS	✓	✓						0
SHARED_POOL_RESERVED_SIZE		✓						X
SHARED_POOL_SIZE		✓						X
SORT_AREA_RETAINED_SIZE		✓						0
SORT_AREA_SIZE		✓		✓				0
SORT_MULTIBLOCK_READ_COUNT				✓				0
TIMED_OS_STATISTICS						✓		0
TIMED_STATISTICS						✓		0

<표 3> 업무별 파라미터 설정 예제

DB_BLOCK_BUFFERS	100	1000
DB_BLOCK_SIZE	2048	8192
LARGE_POOL_SIZE	500	5000
LOG_BUFFER	50000	5000
ROLLBACK_SEGMENTS	30000	300
SHARED_POOL_SIZE	8000	4000
SORT_AREA_SIZE	3000	30000

2.3 파라미터 조정

파라미터를 조정한다는 것은 현재 사용하는 응용 프로그램들이 사용하는 자원(Resource)들의 상태를 파악하여 현재 운용중인 데이터베이스 시스템이 최적의 조건하에서 운영되도록 하기 위하여 수행해야 하는 작업으로써 주로 동적 파라미터의 조정을 의미하는 것이다. 그러나 대개의 데이터베이스 관련자들은 데이터베이스 관리 시스템 판매 업체들이 제공하는 기본 값(Default Value)을 그대로 사용하는 경우가 많으며, 한번 설정된 데이터베이스 파라미터 값들은 변경 없이 그대로 사용하고 있는 것이 현실이다. 물론 데이터베이스 관리 시스템 판매 업체들이 제공한 기본 값들을 사용하는 것이 데이터베이스 시스템 운영에 있어서 위험 부담을 줄일 수 있다는 장점을 갖을 수는 있으나, 최적의 운영 조건이 된다고 할 수는 없을 것이다. 따라서 데이터베이스 파라미터는 처리하고자 하는 응용 업무별로 서로 다른 조건 값을 갖을 수 있으며 또한, 운영되는 응용 프로그램들의 운영 주기 등에 따라서도 파라미터 값은 변경하여 사용될 수 있다[4]. 예를 들어 변동처리(Transaction) 처리가 많이 발생하는 온행업무 시스템과, 한번 저장된 자료를 자주 참조하는 업무가 대부분인 기업의 인사관리 업무 시스템은 파라미터 값에서 서로 다르게 처리되어야 하며 관리해야 하는 파라미터도 다를 것이다. <표 3>은 두 가지 업무에 있어서 업무 특성을 고려하여 서로 다르게 설정되어야 하는 파라미터의 예제이다.

변동처리가 많은 은행업무는 인사관리 업무에 비교하여 상대적으로 메모리는 적게 필요하지만 프로세스는 많이 필요하다는 것을 알 수 있다. 아울러 변동처리가 많은 은행업무는 로그(Log)를 저장하는 공간이 더 필요하다는 것을 알 수 있다. 따라서 데이터베이스 사용자들은 자신의 업무 특성을 이해한 파라미터 값 지정을 수행해야 할 것이다. 이러한 업무의 특성에 따라 정의되는 파라미터들은 한번 지정하면 그대로 지속적으로 사용하는 것이 대부분이지만 업무의 운영 주기별로 파라미터를 변경할 필요가 있다. 예를 들어 변동처리가 많이 발생하는 은행업무의 예제에서 월말에 한달 동안 처리된 모든 변동 처리 데이터를 1개월에 한번 집계를 하고 분석을 하는 업무를 수행하고 또 다른 업무에 반영한다면, 이러한 작업을 수행할 때에는 인사관리 업무 수행 시에 지정했던 파라미터 값과 같이 변경하여

처리하는 것이 훨씬 좋은 성능을 보장 받을 수 있다는 것을 알 수 있다. 아울러 하루 중에도 시간대별로 처리하는 업무나 업무의 성격이 다르다면 해당 업무 수행 시에 파라미터 값을 변동시키는 것을 신중히 고려해볼 필요가 있다. 따라서 파라미터의 변경은 해당 업무의 특성에 맞추어 설정해야 하며, 업무의 성격 및 처리 주기 등을 고려한 변경을 수시로 해야 한다는 것을 알 수 있다[11].

3. 데이터베이스 파라미터 튜닝 프로세스

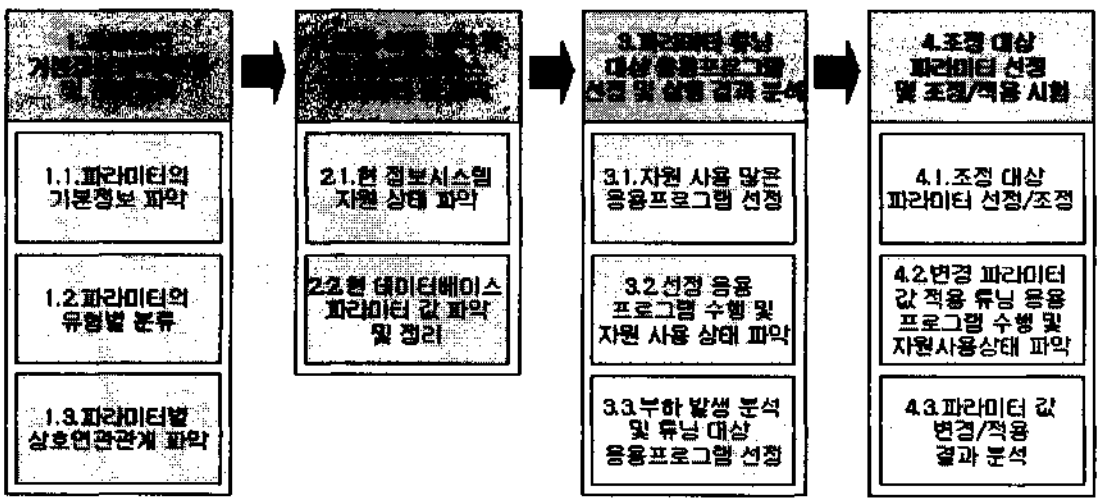
3.1 데이터베이스 파라미터 튜닝 단계

데이터베이스 파라미터 튜닝은 기존 데이터베이스의 응용 프로그램 목록을 기반으로 자원 사용이 많은 프로그램 선정을 시작해서 프로그램 실행 결과 자원 사용 상태를 파악하고 이를 분석하여 조정 대상 파라미터 선정 작업을 실행하고 그 대상 파라미터와 연관된 파라미터를 선정한다. 선정 된 파라미터의 값을 변경 및 조정 작업을 거쳐 만족스러운 성능 향상의 결과가 나오게 되면 파라미터 튜닝 작업을 완료하고 그렇지 못한 경우 다시 프로그램 실행 결과 분석 작업으로 돌아가서 최적의 성능을 발휘 할 때 까지 반복 수행한다.

3.2 데이터베이스 파라미터 튜닝 프로세스

3.2.1 파라미터 기본 정보 파악/분류 및 관계 정의

파라미터 기본 정보 파악/분류 및 관계정의에서는 해당 데이터베이스와 관련된 파라미터의 기본 정보들에 대하여 파악하고 정리,



<그림 3> 데이터베이스 파라미터 튜닝 단계도

분류한다. 그리고 해당 파라미터와 연관 있는 또 다른 파라미터들에 대해서도 정리한다. <그림 4>는 파라미터 기본 정보 파악/분류 및 관계 정의 단계에서 수행 해야 할 활동의 연관 관계를 표현한 것이다. 이 단계에서는 다음의 세가지 활동이 포함된다.

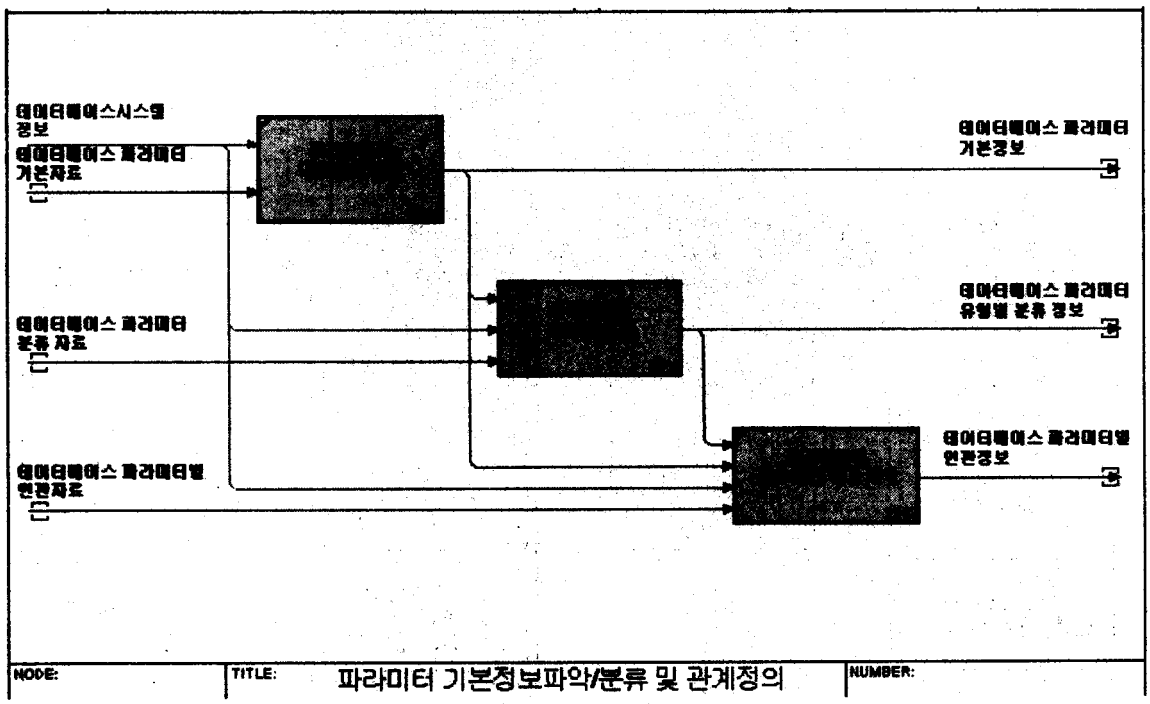
① 파라미터의 기본 정보 파악
② 파라미터의 유형별 분류
③ 파라미터별 상호 연관 관계 파악

파라미터 기본 정보 파악/분류 및 관계 정의에서 필요로 하는 기본 자료는 대부분 데이터베이스 업체에서 제공하는 관리자 메뉴얼(Administration Manual)에 나타나 있다.

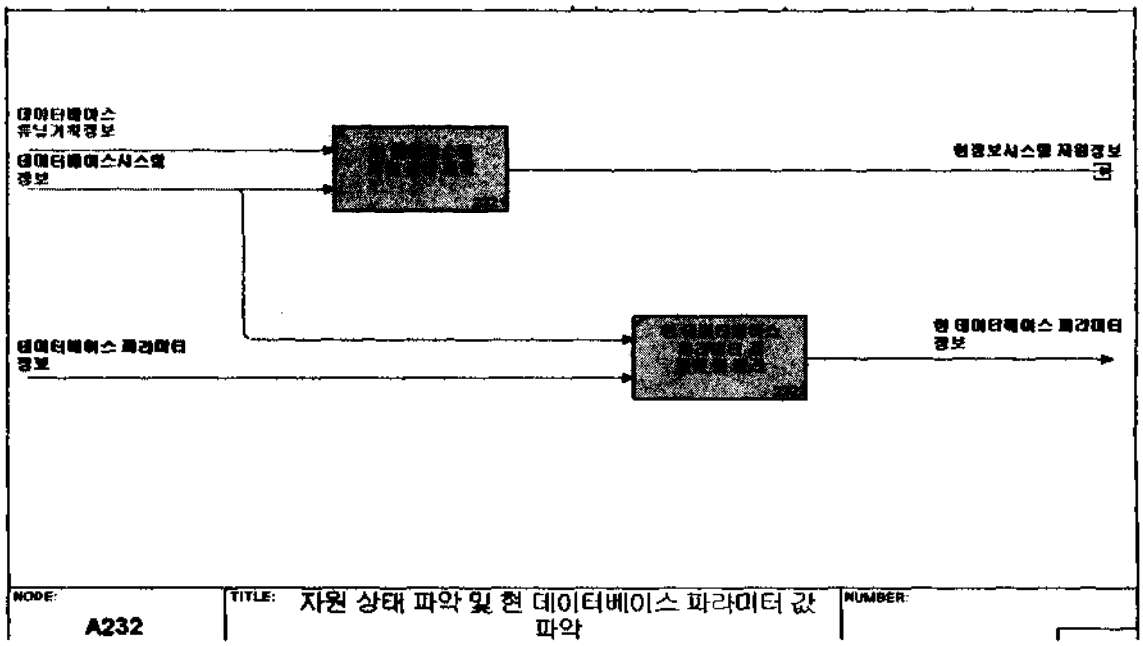
3.2.2 자원 상태 파악 및 현 데이터베이스 파라미터 값 파악

자원 상태 파악 및 현 데이터베이스 파라미터 값 파악 단계에서는 데이터베이스 튜닝 기획 단계의 하드웨어 시스템 구조 정의에서 파악한 현재의 하드웨어 시스템에 대한 자료와 함께 현재의 데이터베이스 시스템이 설치된 내역에 대한 자료를 기준으로 현 정보 시스템 자원 상태 파악을 수행한다. 다음으로는 현재 운영 중인 데이터베이스의 파라미터 값을 정리하여 다음 단계에서 참조 될 수 있도록 한다.

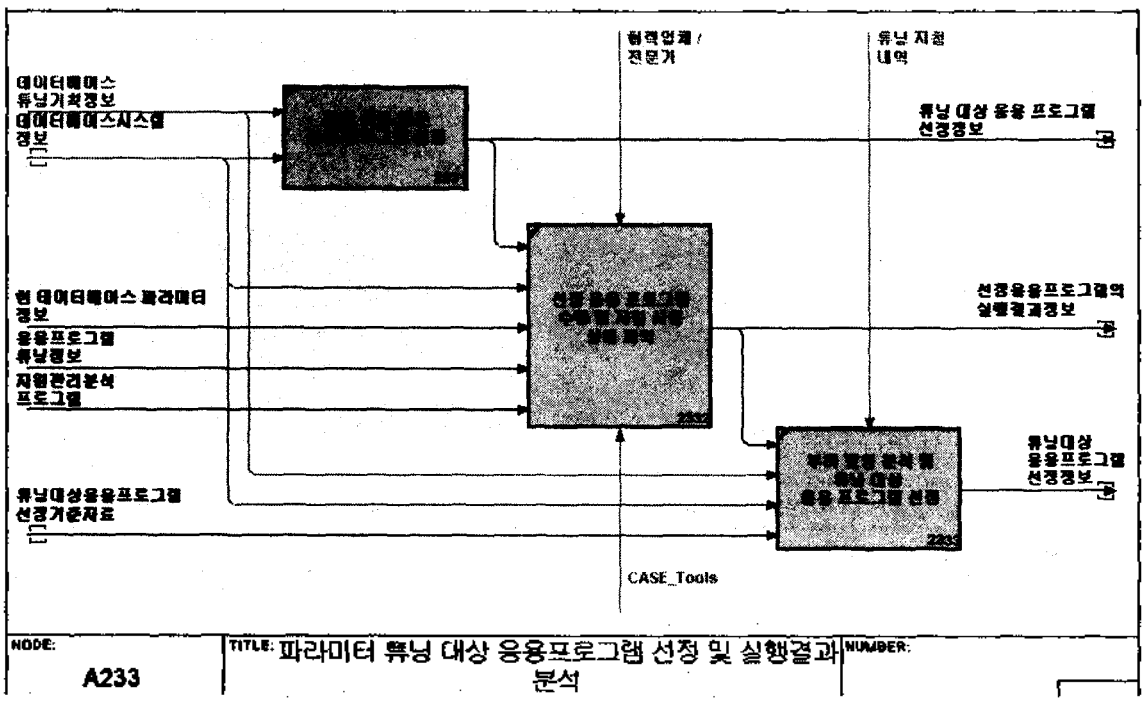
① 현 정보 시스템 자원 상태 파악
② 현 데이터베이스 파라미터 값 파악 및 정리



<그림 4> 파라미터 기본 정보 파악/분류 및 관계 정의



<그림 5> 자원 상태 파악 및 현 데이터베이스 파라미터 값 파악



<그림 6> 파라미터 튜닝 대상 응용 프로그램 선정 및 실행 결과 분석

3.2.3 파라미터 튜닝 대상 응용 프로그램 선정 및 실행 결과 분석

파라미터 튜닝 대상 응용 프로그램 선정 및 실행 결과 분석 단계는 다음의 3 가지 활동이 포함 된다[7].

① 자원 사용 많은 응용 프로그램 선정
② 선정 응용 프로그램 수행 및 자원 사용 상태 파악
③ 부하 발생 분석 및 튜닝 대상 응용 프로그램 선정

자원 사용 많은 응용 프로그램 선정 활동에서는 현재의 데이터베이스 시스템에서 사용하는 응용 프로그램은 물론 저장 프로시져, 트리거, 패키지, 뷰 등 모든 프로그램의 목록을 포함한 데이터베이스 시스템 정보와 데이터베이스 튜닝 기획 정보 단계에서 기획한 튜닝 관련 기대치 등에 대한 자료를 근거로 대용량의 데이터를 사용하거나 다수의 조인이 발생하는 프로그램들 또는 성능 향상을 기대하는 프로그램 등을 선정한다.

선정 응용 프로그램 수행 및 자원 사용 상태 파악 활동에서는 선정된 응용 프로그램을 실제로 수행 시켜서 프로그램 수행 시에 활용하는 자원의 상태 등을 파악한다. 이 단계에서는 성능 측정을 용이하게 해주며, 시스템의 성능에 대한 지표 정보를 손쉽게 수집해주는 성능 측정용 케이스 툴을 실행 시킨 상태에서 성능 측정 대상 응용 프로그램을 실행한다. 그리고 응용 프로그램을 실행하며 발생하는 시스템 자원들에 대한 부하를 측정하는 방법으로 데이터베이스 파라미터 조정 전 자원 사용 상태를 파악한다. 성능 측정용 툴들로부터 얻어진 시스템에 발생된 부하 상태를 수집한다. 수집되는 정보는 CPU 사용, 메모리 사용, 디스크 I/O, 자원 경합 등의 자료들 이다. 여기서 수집된 정보를 분석하여 시스템 자원 사용 상태를 파악한다. 이 활동에서는 케이스 도구의 도움 뿐 아니라, 협력업체 또는 데이터베이스 튜닝 관련 전문가 들의 도움을 받아 같이 작업할 것을 권고한다. 그리고 이 활동에서 사용하는 자원 관리 분석 프로그램은 데이터베이스 관리 시스템 공급업체에서 제공한다. 만일 제공된 관리 프로그램이 없다면 간단히 작성하여 활용할 수 있다. <표 4>는 ORACLE에서 활용 할 수 있는 파라미터 튜닝 관련 관리 프로그램의 예제로써 파라미터 변경 지침이 포함된 것이다.

<표 4> ORACLE 파라미터 튜닝 관리 프로그램 예제

```
rem ************************************************************
rem *          system으로 connect 되어야 수행가능하다.          *
rem *           (자동으로 sysstat.out로 spooling 된다)          *
rem ************************************************************
set feedback offclear columns;
spool instance.lst

prompt ********************************************************
prompt *                LIBRARY CACHE TUNING
prompt ********************************************************
select to_char(trunc(sum(reloads)/sum(pins)*100, 5),99.99999)||'%(less than 1%)' "Library Cache MISS RATIO" from
v$librarycache;
```

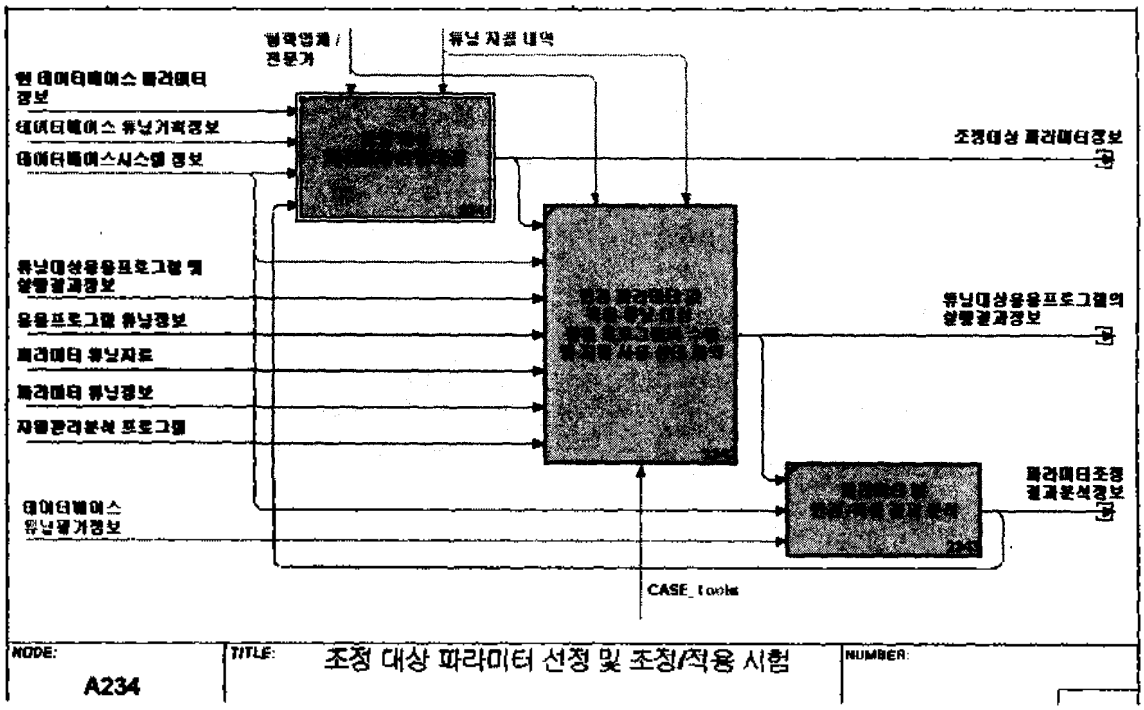
```
prompt
prompt #  libary cache miss ratio is good if it is less than 1 -2 %
prompt #  ratio를 늘리려면 shared pool size의 값을 증가 시켜야 한다.
pause ... please press enter key ....

prompt
prompt ******************************************************
prompt *              DICTIONARY CACHE TUNING                *
prompt ******************************************************
select trunc(sum(getmisses)/sum(gets)*100, 5)||'%  (less than 9.8%)' "Data dictionary miss ratio "from v$rowcache;
prompt  #  data dictionary miss ratio is good if it is less than 9.8%
prompt  #  ratio를 늘리려면 shared pool size의 값을 증가 시켜야 한다.
pause ... please press enter key ....

col pr_v new_value pr;
col dg_v new_value dg;
col cg_v new_value cg;
col bw_v new_value bw;
select value pr_v from v$sysstat where name = 'physical reads';
select value dg_v from v$sysstat where name = 'db block gets';
select value cg_v from v$sysstat where name = 'consistent gets';
select total_waits bw_v from v$system_event where event='buffer busy waits';

prompt ******************************************************
prompt *                BUFFER CACHE TUNING                  *
prompt ******************************************************
select trunc((1-(&pr/(&dg+&cg)))*100,5) Hitratio from dual;
prompt # buffer_cache hit ratio is good if it is more than 60-70%
prompt # ratio를 늘리려면 parameter file의 db_buffer_blocks의 값을
prompt # 증가 시켜야 한다.
pause ... please press enter key ....

prompt ******************************************************
prompt *                Buffer Busy Wait Ratio               *
prompt ******************************************************
select trunc((1-(&bw/(&dg+&cg)))*100,5) Hitratio from dual;
prompt # 80%이하이면 Rollback Segment또는 Data Block에서의
prompt # Contention이 심한경우로 v$waitstat의 정보를 토대로
prompt # Rollback Segment를 추가해 주거나 동시 사용자가 많은
prompt # Table의 Free List개수를 늘려준다.
pause ... please press enter key ....

prompt ******************************************************
prompt *              FREE LIST'S WAIT RATIO                 *
prompt ******************************************************
select trunc(a.count/(&dg + &pr)*100, 5)||
'%' "Free list wait ratio(1%이하)" from v$waitstat awhere a.class = 'free list';
prompt # Freelists는 insert나 update시 검색되는 해당 object의
prompt # free block list로서 동시에 access될 때 Performance에 영향을
prompt # 미치는 중요한 요소이며 특히 OPS환경에서 더욱 유의해야 한다.
prompt #  (Db Block Gets + Consistents Gets) < 1%
pause ... please press enter key ....

prompt ******************************************************
prompt *        INTERNAL SORT AND EXTERNAL SORT              *
prompt ******************************************************
select name, value from v$sysstatwhere name in ('sorts (memory)', 'sorts (disk)');
prompt # MEMORY SORT에 비해 DISK SORT가 상대적으로 많으면
prompt # SORT_AREA_SIZE의 크기를 늘려 주거나 또는
```

```
prompt # Sort_area_retain_size를 적절히 조절하여 Disk I/O를
prompt # 감소시킨다.기본적으로는 on-line환경에서 sort_area_size는
prompt # sort_area_retain_size의 2배정도를 제안하지만 업무에
prompt # 따라 크게 조정할 수 있다.
pause ... please press enter key ....
spool off
```

3.2.4 조정 대상 파라미터 선정 및 조정/적용 시험

조정 대상 파라미터 선정 및 조정/적용 시험 단계에서는 튜닝 대상 응용 프로그램이 실행되는 동안 수집된 시스템 자원 사용 상태 정보를 토대로 CPU 사용, 메모리 사용, 디스크 I/O 회수, 자원 경합 사용 등을 분석하여 현 데이터베이스 파라미터 정보와 비교하여 성능 저하에 영향을 미치는 조정 대상 파라미터를 선정한다. 이 단계에서는 수행하는 CPU 사용 분석 절차는 모든 세션에 대한 CPU 사용 정보를 분석하고 SQL문의 재 파싱(parsing), 부적절한 SQL 문장, 읽기 일관성, 응용 프로그램의 범위 제약, 래치 경합 등의 요소 사항을 분석한다. 그리고 메모리 사용 분석 절차는 시스템의 전체 메모리 양, 공유 풀 사이즈(Shared Pool Size), 버퍼 캐쉬(Buffer Cache), 리두 로그 버퍼(Redo Log Buffer), 정렬 공간(Sort Area) 등의 사용 상태를 분석한다. 디스크 I/O 분석 절차는 물리적인 읽기/쓰기 회수, 논리적인 읽기/쓰기 회수, 읽기/쓰리를 위한 전체 디스크 I/O 회수 등을 분석한다. 그리고 디스크를 이용한 정렬 정도, 체크포인트(Checkpoint)회수 등을 분석한다. 자원 경합 분석 절차는 전체 자원의 현재 활용도와 최대 가용 활용도에 대한 정보를 분석하고 자원을 사용하기 위해 대기하는 시간과 회수를 분석한다. 그리고 롤백 세그먼트(Rollback Segment), 다중스레드 서버(Multithreaded Server) 프로세스, 병렬 서버 프로세스, 리두 로그 버퍼 래치(Redo Log Buffer Latch), LRU 래치, Free List 등에 대한 자원 경합을 분석한다[8].

다음으로는 변경 대상 파라미터에 대한 새로운 파라미터 값을 제시하고, 이를 적용하여 현 파라미터 값을 변경하여 데이터베이스 인스턴스를 다시 기동 시켜 그 변경 사항이 적용 되도록 한다. 즉, 파라미터 값의 변경이 끝나면 데이터베이스 인스턴스를 정지(shutdown) 시키고 다시 기동(startup) 시켜서 변경된 파라미터의 값이 새로 기동되는 데이터베이스 인스턴스에 반영 되도록 한다. 그리고, 변경된 파라미터를 기준으로 튜닝 대상 응용 프로그램을 다시 수행시켜 성능 변화를 측정하고, 사용된 자원을 파악한다. 이렇게 파라미터 변경으로 변화된 성능 변화를 파악 하고 분석하여 만족할 만한 수준인가를 평가한다. 여기서 만족스러운 성능 향상이 이루어지게 되면 파라미터 튜닝 단계를 완료하게 되고, 성능 향상 결과에 만족하지 못하면 이 단계들을 반복 수행하게 된다.



<그림 7> 조정 대상 파라미터 선정 및 조정/적용 시험

다음의 2가지 예제는 ORACLE 8i 데이터베이스 관리 시스템의 파라미터를 조정하여 성능 변화가 생기는 결과를 제시하는 것이다. 이 예제들 중의 하나는 데이터베이스 파라미터 중에서 메모리 활용에 이용되는 가장 중요한 'DB_BLOCK_BUFFERS' 파라미터 값을 변경한 예제이고, 다른 하나는 데이터를 정렬하기 위해 활용하는 'SORT_AREA_SIZE' 파라미터 값을 변경하는 예제이다. 이 테스트에 사용한 데이터는 기본 식별자를 포함하여 10010 Byte의 크기이며, 32764건의 Row로 구성되어있는 테이블을 대상으로 했으며, 테스트 환경의 하드웨어는 HP Pentium PC Server이고, 프로세서는 900MHz짜리 한 개이며, 메인 메모리는 512 메가였다. 먼저 'DB_BLOCK_BUFFERS' 파라미터 값을 변경한 예제는 32764건의 전체 Row에 대해 전체 테이블 스캔 (Full Table Scan)을 수행(SELECT * FROM TEST)한 결과를 제시한 것이다. 다음으로 'SORT_AREA_SIZE' 파라미터 값을 변경하는 예제는 32764건의 전체 Row에 대해 특정 Column들을 대상으로 정렬하는 경우와 조건을 주어 필요한 Row를 선택하여 정렬을 수행한 결과의 예제이다. 그리고 모든 테스트의 각 결과는 해당 조건에 대해 5회 이상을 수행하여 평균 값을 나타낸 것이다.

<표 5>는 DB_BLOCK_BUFFERS 파라미터를 50 ~ 19200으로 변환하여 공유 메모리인 SGA의 크기가 변환하는 것을 표로 정

리한 것이다. 그리고 <그림 8>은 측정 항목들 중에서 파라미터 변화가 결과에 차이를 나타낸 몇 개의 측정 항목에 대하여 정리한 것으로써 변화를 알아보기 쉽도록 그래프화한 것이다.

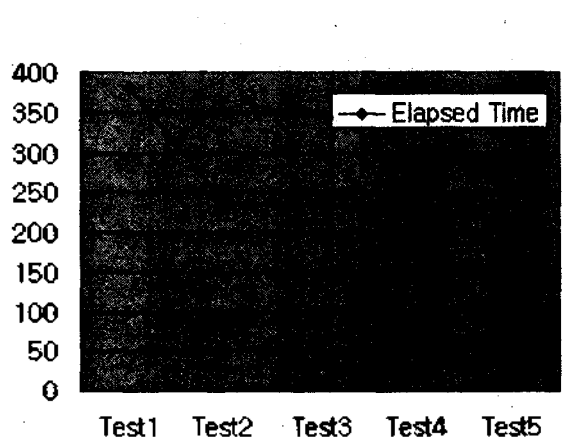
결과를 살펴보면 파라미터 값을 무조건 크게 하여 사용되는 자원을 많이 할당한다고, 이에 비례하여 성능이 많이 증가하는 것이 아니라 어느 정도의 증가를 보이다 변화가 거의 발생하지 않는 것을 알 수 있다. 따라서 데이터베이스 파라미터 튜닝에서는 최적의 파라미터 값을 파악하기 위한 수행을

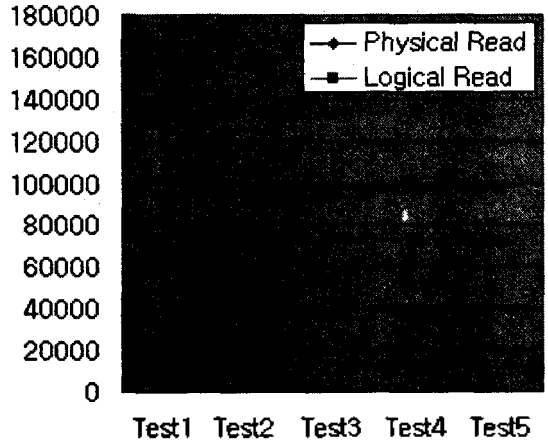
<표 5> 파라미터 변경 조건 예제 - 'DB_BLOCK_BUFFERS'

DB_BLOCK_SIZE : 8196 Bytes

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Total SGA
Test 1	50	50 MB	400 KB	600 KB	71.08011 MB
Test 2	100	50 MB	800 KB	600 KB	71.47073 MB
Test 3	1024	50 MB	8 MB	600 KB	78.68948 MB
Test 4	4096	50 MB	32 MB	600 KB	102.68948 MB
Test 5	19200	50 MB	150 MB	600 KB	220.68948 MB

	Elapsed Time	Physical Read	Logical Read
Test1	380,31	65651	163830
Test2	234,3	73723	98302
Test3	233,82	73672	98302
Test4	231,46	73415	98302
Test5	216,23	71665	98302





<그림 8> 파라미터 변경 수행 결과 예제 - 'DB_BLOCK_BUFFERS'

반복적으로 실행해서 최적의 값을 도출해야 한다는 것을 알 수 있다. 본 예제의 경우 DB_BLOCK_BUFFERS의 숫자를 50에서 100으로 증가 했을 때 자원 할당, 즉 메인 메모리는 400K Byte 정도 더 할당되고, 약 40 % 정도의 CPU Elapsed Time의 감소를 보였다. 그러나 100에서 1024 개로 증가를 했을 경우나, 1024에서 4096, 19200으로 증가를 했어도 성능의 변화는 거의 없는 것으로 나타났다. 아울러 활용한 메모리에 발생한 Input/Output을 나타내는 Physical /Logical Read에 대한 결과도 거의 유사하게 결과가 도출되었다. 따라서 이러한 경우에는 DB_BLOCK_BUFFERS의 숫자를 100 정도로 하는 것이 가장 좋은 결정이 되는 것이다. 그 이유는 데이터베이스 버퍼 블록만 크다고 더 많은 데이터가 한꺼번에 처리되는 것이 아니라 디스크 I/O와 프로세스의 처리 능력 등에 따라 달라지기 때문이다. 물론 이 결과도 해당 응용 프로그램이 수행되는 같은 시간대에 메모리를 활용하는 또 다른 응용프로그램이 존재한다면 그 결과는 달라질 것이다.

<표 7>의 두 가지 시험 예제 중 Test1은 데이터 Row 32764건 데이터 전체를 테이블 스캔 하는 예제 (SELECT * FROM TEST ORDER BY B;)이고, Test2는 데이터 Row 32764건 중에서 3640건을 질의하여 정렬하는 예제 (SELECT * FROM TEST WHERE A LIKE '1%' ORDER BY B;)를 수행하기 위한 조건을 구분하여 만든 것이다.

<표 6> 파라미터 변경 조건 예제 - 'SORT_AREA_SIZE'

Test 1 : DB_BLOCK_SIZE : 8196 Bytes

Test 1			Buffer Cache	Large Pool	Total SGA	Sort_Area_Size
Test 1-1	50	50 MB	150 MB	600 KB	220.68948 MB	64 KB
Test 1-2	50	50 MB	150 MB	600 KB	220.68948 MB	128 KB
Test 1-3	50	50 MB	150 MB	600 KB	220.68948 MB	512 KB
Test 1-4	50	50 MB	150 MB	600 KB	220.68948 MB	1024 KB
Test 1-5	50	50 MB	150 MB	600 KB	220.68948 MB	10 MB
Test 1-6	50	50 MB	150 MB	600 KB	220.68948 MB	20 MB
Test 1-7	50	50 MB	150 MB	600 KB	220.68948 MB	50 MB
Test 1-8	50	50 MB	150 MB	600 KB	220.68948 MB	100 MB
Test 1-9	50	50 MB	150 MB	600 KB	220.68948 MB	200 MB

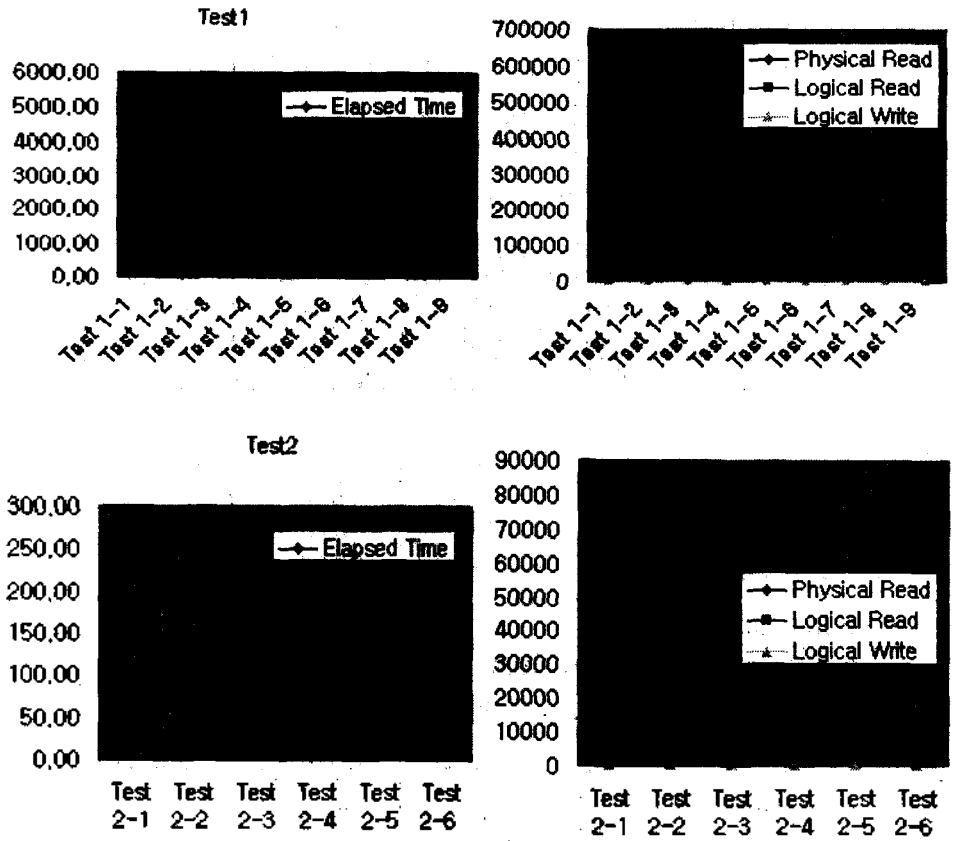
Test 2 : DB_BLOCK_SIZE : 8196 Bytes

Test 2			Buffer Cache	Large Pool	Total SGA	Sort_Area_Size
Test 2-1	50	50 MB	152 MB	600 KB	222.68948 MB	1024 KB
Test 2-2	50	50 MB	152 MB	600 KB	222.68948 MB	10 MB
Test 2-3	50	50 MB	152 MB	600 KB	222.68948 MB	25 MB
Test 2-4	50	50 MB	152 MB	600 KB	222.68948 MB	50 MB
Test 2-5	50	50 MB	152 MB	600 KB	222.68948 MB	70 MB
Test 2-6	50	50 MB	152 MB	600 KB	222.68948 MB	100 MB

<표 7> 파라미터 변경 조건 예제 결과-'SORT_AREA_SIZE'

Test 1	Elapsed Time	Physical Read	Logical Read	Logical Write
Test 1-1	5455.60	602616	125991	514
Test 1-2	4891.31	643378	111933	532
Test 1-3	2273.81	325591	101595	528
Test 1-4	1682.01	267011	100618	530
Test 1-5	820.71	202689	99704	529
Test 1-6	759.68	202689	99676	529
Test 1-7	739.32	202689	99657	528
Test 1-8	730.79	202689	99651	528
Test 1-9	745.72	202687	99650	529

Test 2	Elapsed Time	Physical Read	Logical Read	Logical Write
Test 2-1	256.24	84780	70319	404
Test 2-2	196.34	80746	70240	403
Test 2-3	194.07	80746	70238	404
Test 2-4	195.43	80744	70236	403
Test 2-5	167.88	73466	70184	382
Test 2-6	168.12	73466	70188	383



<그림 9> 파라미터 변경 수행 결과 예제 - 'SORT_AREA_SIZE'

<그림 9>는 Test1과 Test2의 수행 결과이다. Test1의 결과는 SORT_AREA_SIZE가 커지면 성능 개선의 폭이 큰 차이를 나타냈다. 그러나 Test1-5 이후의 SORT_AREA_SIZE 20M 이상의 경우에는 차이가 별로 없었다. 그 이유는 정렬 처리하는 데이터가 32764건 밖에 안되기 때문이다. 만일에 정렬되는 데이터가 더 많았다면 그 차이는 더욱 크게 벌어질 것이다. Test2의 경우 즉 3640건의 데이터를 정렬하는 경우에는 SORT_AREA_SIZE가 변하더라도 그 수행 결과 값은 별 차이를 보이지 않는다. 이는 적은 데이터의 처리는 메모리 정렬을 수행하기 때문에 SORT_AREA_SIZE의 크기 변화에 따른 차이가 별로 없는 것이다. 따라서 데이터베이스 시스템 관련자들은 이러한 파라미터들의 특성 및 적용 방법 등을 정확히 파악하여 자신의 데이터베이스 시스템 운영 환경에 최적의 파라미터를 설정하여야 할 것이다.

4. 결론

데이터베이스 튜닝은 정보 시스템이 구축 중이거나, 이미 구축되어 활용 중인 데이터베이스 시스템에 대하여 보다 확실한 성능 향상을 보장하기 위한 것으로써 데이터베이스 모델 튜닝, SQL문장 튜닝, 파라미터 튜닝, 시스템 튜닝에 이르는 여러 단계로 나누어 질 수 있으며 해당 튜닝 담당자들도 서로 달라 질 수 있다. 데이터베이스 튜닝은 데이터베이스 설계 단계에서 시작하는 것이 가장 적절하겠으나, 데이터베이스 운영 사정에 따라 각 튜닝 대상과 튜닝 단계를 선택하여 실시 할 수 있다. 그러나 현재까지의 데이터베이스 튜닝 작업은 단순히 경험과 정보 시스템 구축 단계별로 부분적인 튜닝을 수행하고 있다[10]. 따라서 소프트웨어 공학 방법론이 적용된 튜닝 프로세스의 정립이 필요한 상태이다. 이에 본 논문에서는 데이터베이스 튜닝 프로세스의 정립에 대하여 설명하였다. 특히 각 데이터베이스 관리 시스템들에 존재하는 수 많은 파라미터들에 대한 분류를 수행하여 정리하는 방법과 파라미터들 간의 연관 관계를 파악하여 정리된 파라미터들을 활용하는 방법 등에 대한 프로세스를 설명했다. 여기서 파라미터들은 메모리, 프로세스, 디스크, 네트워크 등에 영향을 미치는 기준으로 분류할 수 있었으며 이렇게 분류함으로써 응용프로그램의 자원 사용 상태를 분석한 결과와 파라미터 분류를 바탕으로 조정이 필요한 파라미터를 선정하는 작업에 도움을 줄 수 있다는 것을 알 수 있었다.

그리고 현재의 파라미터 값들을 파악하고 변경 대상 파라미터를 설정하는 방법등에 대해 설명했으며 아울러 파라미터의 변경 주기 등을 고려한 파라미터의 선정/조정/적용에 대한 설명을 했다. 특히 파라미터 변경 관련 예제를 들어 적절한 파라미터 변경의 필요성을 강조했다.

또한 파라미터는 한번 적용하여 지속적으로 사용하는 것이 아니라 업무 기능, 환경 변화에 따라 변경을 해 가면서 사용하는 것이 보다 나은 데이터베이스 시스템 운영을 수행 할 수 있다는 것을 설명하였다.

본 논문에서는 데이터베이스 파라미터 튜닝 단계의 프로세스를 제시 함으로써 체

계화된 절차에 따라 데이터베이스 파라미터 튜닝 작업을 수행 할 수 있다는 것을 보였으며 데이터베이스 파라미터 튜닝 단계의 프로세스에 따라 튜닝 작업을 수행 함으로써 무수히 많은 파라미터에 대한 튜닝 작업을 체계적으로 수행 할 수 있고, 결과적으로 데이터베이스 시스템을 항상 최적의 성능을 발휘 할 수 있도록 유지할 수 있다는 것을 설명하였다.

참고 문헌

[1] Dennis Shasha, "Tuning Database for High Performance", ACM Computing Surveys, Vol.28, No.1, pp.113-pp.5, March 1996
[2] Dennis E. Shasha, Database Tuning, Prentice Hall, 1992
[3] James Martin, Information Engineering, Prentice Hall, 1990
[4] Abraham Silberschatz et al, Database System Concepts, McGraw-Hill, 1997
[5] Oracle 8 : Database Administration, 성능튜닝 워크숍, SQL 튜닝, ORACLE, 1998
[6] Oracle 8i: Designing and Tuning for Performance, ORACLE, 1999
[7] Heidi Thorpe, Oracle8i Tuning and Administration, Addison-Wesley, 2001
[8] Carma McClure, The Three R's of Software Automation, Prentice Hall, 1992
[9] http://dblab.changwon.ac.kr/, Changwon National University, 1996
[10] 최용락, "튜닝을 통한 데이터베이스 시스템 성능 향상", 1996
[11] 최용락, "데이터 모델링 실무", 드림 미디어, 2000

저자 소개

최용락(e_mail : ylch@hitel.net)
숭실대학교 전자계산학과 졸업(공학 학사)
숭실대학교 정보과학대학원 정보산업학과 졸업(이학 석사)
숭실대학교 대학원 컴퓨터학과 박사 수료
2001.03 – 현재 : 숭실대학교 컴퓨터학부 겸임교수
관심 분야 : 데이터베이스, ERP, 소프트웨어공학

윤병권(e_mail : superman@it.soongsil.ac.kr)
숭실대학교 전자계산학과 졸업(공학 학사)
숭실대학교 대학원 컴퓨터학과 석사 졸업
숭실대학교 대학원 컴퓨터학과 박사 과정
관심 분야 : 소프트웨어공학, 소프트웨어 개발 방법론

정기원(e_mail : chong@computing.soongsil.ac.kr)
서울대학교 공과대학 전기공학과
알라바마 주립대학(헌츠빌) 전산학과 전산학석사
텍사스 주립대학(알링턴) 전산학과 전산학박사
1990.03 – 현재 : 숭실대학교 컴퓨터학부 교수
관심 분야 : 소프트웨어 개발 방법론, 소프트웨어 프로세스, 품질 보증, 감리, CALS/EC, 실시간 시스템, 모델링 및 시뮬레이션