4장. DB2 Monitoring 및 Tuning

4.1 DB2 엔진의 구조

4.2 Database Manager Configuration Parameter

4.3 Database Configuration Parameter

4.4 스냅샵 모니터링

4.5 Event Monitor

4.6 Snap Shot 내용 분석(참고 자료)

4.3.1 Buffer Pool Size
4.3.2 Sort
4.3.3 Lock

4.7 성능 모니터

4.8 성능 구성 스마트 가이드

4.9 SQL Access Plan 분석 툴

4.10 Index Adivisor

4.11 DB2 조정자(governor)

4.1 DB2 엔진의 구조

DB2에서의 아키텍처의 주요 특성은 데이터베이스 무결성을 보장하는 능력이다. 주요한 데이터베이스 리소스(Resources)로부터 모든 데이터베이스 응용프로그램을 독립시킨다. 이 리소스들은 데이터베이스 제어 블록과 주요한 데이터베이스 파일들이다.

또한, 데이터베이스 연결 과정동안 DB2 조정 에이전트는 각 데이터베이스 응용프로그램에 할당된다. 각각의 DB2 에이전트는 데이터베이스 응용프로그램을 대신하여 작업하고 모든 SQL 요청들을 다룬다. 응용프로그램과 데이터베이스 에이전트는 IPC(Inter Process Communication) 기술(메시지 큐, 공유 메모리, 세마포아 등)을 이용하여 통신한다. DB2 코디네이터 에이전트는 파티션내 병렬 처리가 가능하다면, DB2 서브 에이전트와 함께 작업한다. 이 아키텍처는 잘못된 응용프로그램으로부터 데이터베이스 자원들을 보호하기 위해 방화벽을 제공한다.

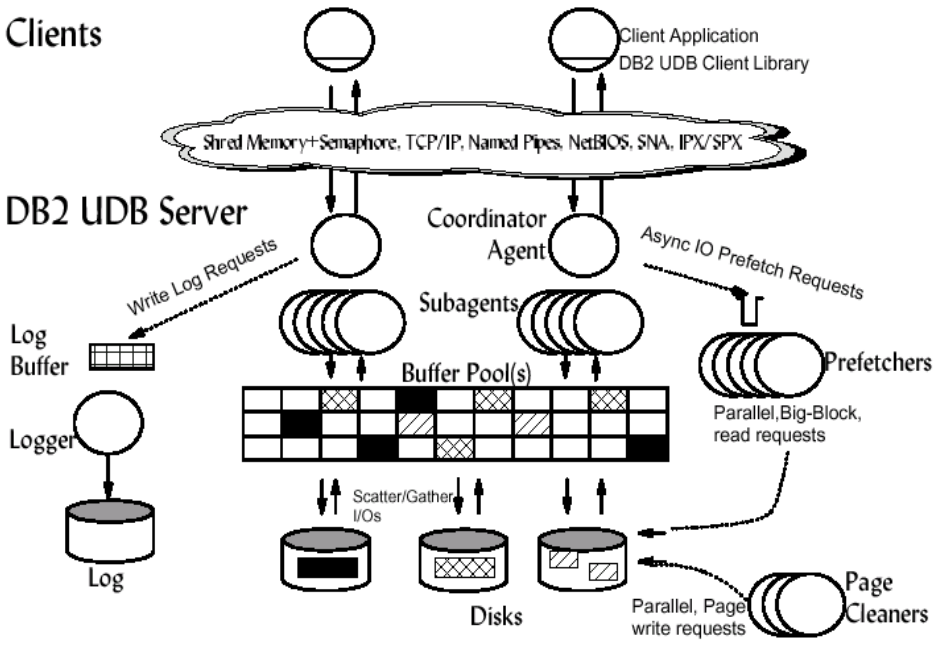


그림 1-1. DB2 프로세스 모델

DB2 에이전트는 Windows와 OS/2에서는 쓰레드(Thread)이나, UNIX 운영체제에서는 Process이다.

각 클라이언트 응용프로그램은 DB2 UDB 클라이언트 라이브러리와 링크되어 있고,

공유 메모리와 세마포아 (지역 클라이언트) 또는 TCP/IP와 APPC와 같은 통신 프로토콜 (원격 클라이언트)을 사용하여 DB2 UDB 서버와 통신한다.

서버 쪽에서는 활동이 EDU(Engine Dispatchable Unit)에 의해 제어된다. EDU는 AIX를 포함한 UNIX에서는 프로세스들로 구현된다.

1) DB2 에이전트

코디네이터 에이전트와 서브 에이전트를 포함한 DB2 에이전트는 응용프로그램을 대신하여 SQL 문장들의 처리를 수행하는 DB2 UDB 프로세스들의 가장 일반적인 유형이다. UNIX에서는 ps 명령어를 사용하면 코디네이터 에이전트 프로세스(db2agent)와 서브 에이전트(db2agntp)를 관찰할 수 있다.

2) 버퍼 풀

버퍼 풀은 사용자 테이블 데이터, 인덱스 데이터, 카탈로그 데이터의 데이터 페이지들이 디스크 저장 공간에서 임시로 이동하는 저장 메모리 영역이다. DB2 에이전트들은 버퍼 풀에서 데이터 페이지들을 읽고 수정한다. 버퍼 풀은 데이터가 디스크에서보다 메모리에서 좀 더 빠르게 액세스되기 때문에 데이터베이스 성능의 주요한 요소이다.

3) 프리페처

프리페처는 응용프로그램이 데이터를 필요로 하기 이전에 디스크에서 데이터를 추출하여 버퍼 풀로 이동하는 역할을 한다. 요청된 페이지들을 디스크에서 버퍼 풀로 가져오기 위해 프리페처는 대형 블록(Big-Block) 또는 스캐터 읽기 입력(Scatter Read Input) 운영방법을 사용함으로써 요청들을 수행한다. UNIX에서는 ps 명령어를 사용하면 프리페처 프로세스(db2pfchr)를 볼 수 있다.

4) 페이지 정리자

페이지 정리자는 프리페처가 디스크 저장 영역에서 페이지들을 읽어 버퍼 풀로 이동하기 전에 버퍼 풀에 방을 만들기 위한 역할을 한다. 예를 들어, 테이블내에 대량의 데이터를 갱신한다면, 버퍼 풀의 데이터 페이지는 변경되나 디스크 저장 영역에는 기록되지 않는다 (이러한 페이지들을 Dirty 페이지라고 함). 페이지 정리자는 응용프로그램 에이전트들과는 관계없이 버퍼 풀에 방이 있다는 것을 보장하기 위해 버퍼 풀에서 페이지들을 찾아 기록한다. UNIX에서는 ps 명령어를 사용하면 페이지 정리자 프로세스

(db2pclnr)를 볼 수 있다.

5) 로그

버퍼 풀내의 데이터 페이지에 대한 변경들이 로깅된다. 데이터베이스내에서 데이터 레코드를 변경하는 에이전트 프로세스들은 버퍼 풀내에 관련된 페이지를 변경하고, 로그 레코드를 로그 버퍼로 기록한다. 로그 버퍼에 기록된 로그 레코드들은 로거(Logger)에 의해 비동기적으로 로그 파일로 기록된다. UNIX에서는 ps 명령어를 사용하면 각 활동중인 데이터베이스에 대해서 로거 프로세스(db2loggr)를 볼 수 있다.

데이터베이스 관리자는 변경을 다시 하기(Redo) 위해 관련된 로그 레코드에서 Redo 정보를 사용한다. 이 매카니즘을 손상 복구(Crash Recovery)라고 한다. 데이터베이스 관리자는 데이터베이스를 재시작할 때 손상 복구를 수행한다.

로그 버퍼내의 데이터는 다음의 경우에 디스크에 강제로 기록된다.

- 일치하는 데이터 페이지가 디스크에 강제로 기록되기 전에(Write-ahead 로깅).
- COMMIT시에. 그룹 확약 개수인 MINCOMMIT 데이터베이스 구성 매개변수에 도달한 후에.
- 로그 버퍼가 가득찰 때.

6) Fenced / Not Fenced 자원

데이터베이스 스토어드 프로시저는 CALL 문장을 사용하여 데이터베이스 응용프로그램에서 호출될 수 있는 동적으로 로드되는 라이브러리이다. 이 라이브러리는 DB2 데이터베이스 서버에 저장되고, Fenced 자원 또는 Not Fenced 자원으로써 실행될 수 있다. Fenced 자원은 데이터베이스 에이전트로부터 분리된 프로세스에서 실행되는 것이고, Not Fenced 자원은 데이터베이스 에이전트로써 같은 프로세스내에 실행된다.

Not Fenced 자원은 내부-프로세스 통신 오버헤드가 적기 때문에 Fenced 자원보다 더 나은 성능을 가지지만, Not Fenced 자원은 만약 제대로 테스트되지 않으면 DB2 제어 블록위에 다시 쓸수 있다.

7) 쿼리 병렬 처리

- 쿼리간 병렬 처리(Inter-Query Parallelism)는 동시에 복수개의 응용프로그램이 데이타베이스를 쿼리할 수 있는 능력을 의미한다. 각 쿼리는 다른 쿼리들과 독립적으로 수행되나, DB2는 동시에 모든 쿼리들을 수행한다. DB2는 항상 이러한 유형의 병렬 처리는 지원해 왔다.

- 쿼리내 병렬 처리(Intra-Query Parallelism)는 파티션내 병렬 처리(Intra-Partition Parallelism) 또는 파티션간 병렬 처리(Inter-Partition Parallelism) 또는 모두를 사용하여 동시에 단일 쿼리의 부분을 처리하는 것을 의미한다. 쿼리내 병렬 처리를 가지고, 단일 복잡한 쿼리는 DB2 옵티마이저에 의해 수행될 수 있고, 병렬로 수행될 수 있도록 여러 조각으로 나누어질 수 있다.

파티션내 병렬 처리(Intra-Partition Parallelism)는 한 쿼리를 복수개의 부분으로 쪼개는 능력을 의미한다. 즉 색인 생성, 데이터베이스 로드, SQL 쿼리와 같은 단일 데이터베이스 작업을 복수개의 부분으로 세분하여, 단일 데이터베이스 파티션내에서 병렬로 실행한다. 파티션내 병렬 처리는 SMP 시스템에 매우 적합하다.

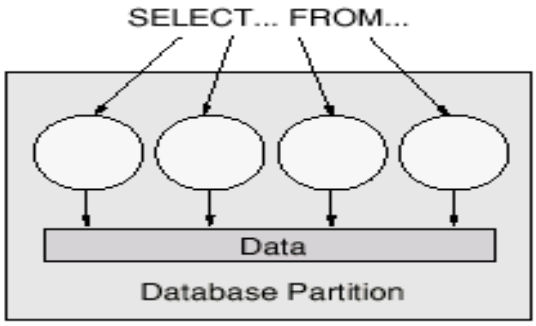


그림 1-2. 파티션내 병렬 처리 (Intra-Partition Parallelism)

- 파티션간 병렬 처리(Inter-Partition Parallelism)

파티션간 병렬 처리는 쿼리를 단일 머신 또는 복수개의 머신에서 분할된 데이터베이스의 복수 파티션 사이로 복수개의 부분으로 쪼개는 능력을 의미한다. 쿼리는 병렬로 수행된다. 파티션간 병렬 처리는 MPP 시스템에 매우 적합하다.

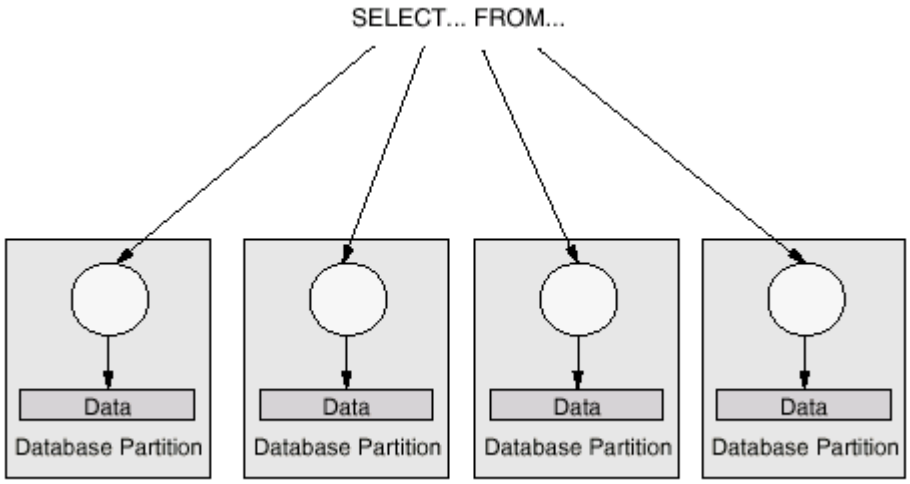


그림 1-3. 파티션간 병렬 처리 (Inter-Partition Parallelism)

그림 1-3에서는 병렬로 수행될 수 있는 4개의 조각으로 쪼개져 결과가 단일 파티션에서 순차적인 방식으로 수행되는 것보다 더 빨리 리턴되는 쿼리를 볼 수 있다. 병렬처리의 정도는 생성한 파티션의 수와 노드 그룹을 정의하는 방법에 따라 대개 결정된다. 파티션간 병렬 처리 뿐만 아니라 파티션내 병렬 처리와 파티션간 병렬처리의 조합이 가능하도록 DB2 EEE (Enterprise-Extended Edition)가 설치될 필요가 있다.

4.2 Database Manager Configuration Parameter

DBMS 구성 파일은 인스턴스(Database Manager) 단위로 존재하며 어플리케이션 수행 환경, 통신 환경, 권한 그룹 등과 같은 인스턴스에 대한 구성 정보를 담고 있다. DB2 관리자는 이러한 DBMS 구성 파일의 매개변수 값들을 조정함으로써 성능을 최적화 할 수 있다.

▷DBM 구성 파일을 보는 방법은 다음과 같다.

```
# login db2inst1
# db2 get database manager configuration
(또는)
# db2 get dbm cfg
```

▷ 매개변수 갱신 방법

```
# db2 update dbm cfg using '매개변수_이름' '값'

<예>
# db2 update dbm cfg using sysadm_group db2iadm1
```

▷ DBM 구성 파일을 변경한 경우 인스턴스를 재시작해야(db2stop & db2start) 변경된 값이 유효해 진다.

▷ DBM 구성 파일 내용 (예)

```
데이타베이스 관리 프로그램 구성 [AIX용 DB2 UDB]


     노드 유형 = 국지 및 원격 클라이언트가 있는 데이타베이스 서버
 데이타베이스 관리 프로그램 구성 릴리스 레벨              = 0x0800
 CPU 속도(밀리초/명령어)                      (CPUSPEED) = 1.006093e-05
 현재 사용 중인 데이타베이스의 최대 수          (NUMDB) = 8
```

```
 트랜잭션 프로세서 모니터명                (TP_MON_NAME) =
 생략시 차지백 계정                      (DFT_ACCOUNT_STR) =
 Java Development Kit 1.1 설치 경로        (JDK11_PATH) =
 진단 오류 캡처 레벨                        (DIAGLEVEL) = 3
 진단  자료  디렉토리  경로                                    (DIAGPATH)  =
/home/db2inst1/sqllib/db2dump

 생략시 데이타베이스 모니터 스위치
   버퍼 풀                              (DFT_MON_BUFPOOL) = OFF
   잠금                                 (DFT_MON_LOCK) = OFF
   정렬(sort)                           (DFT_MON_SORT) = OFF
   명령문                               (DFT_MON_STMT) = OFF
   테이블                               (DFT_MON_TABLE) = OFF
   작업 단위(UOW)                       (DFT_MON_UOW) = OFF
SYSADM 그룹명                           (SYSADM_GROUP) = DB2IADM1
 SYSCTRL 그룹명                         (SYSCTRL_GROUP) =
 SYSMAINT 그룹명                        (SYSMAINT_GROUP) =

 데이타베이스 관리 프로그램 인증          (AUTHENTICATION) = SERVER
 모든 클라이언트 신임                    (TRUST_ALLCLNTS) = YES
 신임 클라이언트 인증                    (TRUST_CLNTAUTH) = CLIENT
생략시 데이타베이스 경로                    (DFTDBPATH) = /home/db2inst1

 데이타베이스 모니터 힙(heap) 크기(4KB)    (MON_HEAP_SZ) = 48
 UDF 공유 메모리 세트 크기(4KB)            (UDF_MEM_SZ) = 256

 백업 버퍼 생략시 크기(4KB)                (BACKBUFSZ) = 1024
 복원 버퍼의 생략시 크기(4KB)              (RESTBUFSZ) = 1024
 정렬 힙(heap) 임계값(4KB)                 (SHEAPTHRES) = 20000
```

```
 디렉토리 캐쉬 지원                                (DIR_CACHE) = YES
 Java 가상 기계 힙(heap) 크기 (4KB)          (JAVA_HEAP_SZ) = 512
적용업무 지원 계층 힙(heap) 크기(4KB)          (ASLHEAPSZ) = 15
 리퀘스터 I/O 블록의 최대 크기(바이트)          (RQRIOBLK) = 32767
 조회 힙(heap) 크기(4KB)                    (QUERY_HEAP_SZ) = 1000
 DRDA 서비스 힙(heap) 크기(4KB)             (DRDA_HEAP_SZ) = 128
 에이전트의 우선순위                              (AGENTPRI) = SYSTEM
 기존 에이전트의 최대 수                          (MAXAGENTS) = 200
 에이전트 풀 크기                           (NUM_POOLAGENTS) = 4 (계산됨)
 풀에 있는 초기 에이전트 수                  (NUM_INITAGENTS) = 0
 조정 에이전트의 최대 수                    (MAX_COORDAGENTS) = MAXAGENTS
 동시 조정 에이전트의 최대 수                    (MAXCAGENTS) = MAX_COORDAGENTS
 DARI 프로세스 유지                               (KEEPDARI) = YES
 DARI 프로세스의 최대 수                          (MAXDARI) = MAX_COORDAGENTS
 색인 재작성 시간                                 (INDEXREC) = RESTART
 트랜잭션 관리 프로그램 데이타베이스명          (TM_DATABASE) = 1ST_CONN
 트랜잭션 재동기화 간격(초)                 (RESYNC_INTERVAL) = 180
 SPM명                                          (SPM_NAME) =
 SPM 로그 크기                              (SPM_LOG_FILE_SZ) = 256
 SPM 재동기화 에이전트 한계                  (SPM_MAX_RESYNC) = 20
 TCP/IP 서비스명                                (SVCENAME) = db2cdb2inst1
 APPC 트랜잭션 프로그램명                         (TPNAME) =
 IPX/SPX 파일 서버명                           (FILESERVER) =
 IPX/SPX DB2 서버 오브젝트명                   (OBJECTNAME) =
 IPX/SPX 소켓 번호                              (IPX_SOCKET) = 879E
 발견 모드                                       (DISCOVER) = SEARCH
 발견 통신 프로토콜                         (DISCOVER_COMM) = TCPIP
 발견 서버 인스턴스                         (DISCOVER_INST) = ENABLE
 디렉토리 서비스 유형                            (DIR_TYPE) = NONE
```

▷ DBM 구성 매개변수 중에서 자주 참조되고 사용되는 것들은 다음과 같다.

(1) Application 관련

가. agentpri : dbm process and threads priority

def : -1

rem : UNIX : low value - high priority

OS/2 : high value - high priority

나. maxagents : 모든 database에 연결하는 application의 총수

(the total number of applications that may connect to all database)

def : 200 (1 - 64000)

rem : concurrent하게 access되는 각 db의 maxappls의 합

다. maxcagents : database manager가 한번에, 동시에 처리하는 dbm agents의 제한(한계)수 (limits the number of dbm agents that can be processed concurrently by the dbm at any one time)

def : -1 (the limit is maxagents 의미)

(2) Agent / Application com Memory

가. aslheapsz : application support layer heap size

(적용업무지원 계층힙 크기)

- comm buffer between local application and its associated agent

def : 15(1 - 524288)

rem : 1) 200

query_heap_sz와 관련, query_heap_sz의 1/5정도

def : 1,000

2) asleapsz > = (size of (input sqlda) + size of

(each input sqlvar) + size of (output sqlda) + 250) / 4096

나. rqrioblk

- client I/O block size (최대 request I/O 블록크기)

- remote db에 connect할 때 client의 comm buffer size

- server에는 initial로 32767 bytes allocate하고, client에 allocate함

다. dos_ rqrioblk : dos requester I/O block

- dos client의 comm buffer size
- remote db access하는 dos/window client에 해당

def : 4096(4096 - 65535)

(3) Agent Private Memory

가. sheapthres : sort heap threshold for dbm

def : UNIX : 4096(250 - 524288)

OS/2 : 2048(250 - 524288)

rem : 각 db에 대한 동시처리되는 agents의 수 * 각 db에 대한 sortheap

-> 모든 db의 합계

(4) Communication

가. TCP/IP 서비스명 (SVCENAME) : TCP/IP service port 이름. 최초 인스턴스 작성시 default 값은 port number 50,000, service name db2cdb2inst1.

나. 발견 모드 (DISCOVER) : Default는 'SEARCH' 이며 클라이언트에서 네트웍 탐색 기능을 이용하여 자동을 서버를 찾아 구성할 수 있도록 지원함

다. 발견 통신 프로토콜 (DISCOVER_COMM) : 네트웍 탐색시 사용하는 프로토콜 설정. Default는 TCP/IP 임

4.3 Database Configuration Parameter

Database 구성 파일은 인스턴스내의 각 데이터베이스에 대한 구성 정보를 담고 있다. 버퍼 풀, 로그 버퍼, 로그 파일, Circular/Archival logging 등 데이터베이스 운영상의 중요한 내용들을 포함하고 있으며, 특히 버퍼 풀과 관련된 매개변수 값들은 데이터베이스의 성능에 직접적인 영향을 미치므로 관리자는 항상 이러한 값들이 적절히 유지될 수 있도록 신경을 써야 한다.

▷ Database 구성 파일을 보는 방법

```
# login db2inst1
# db2 get database configuration for ‘db_name’
(또는)
# db2 get db cfg for ‘db_name’
```

▷ Database 구성 파일의 매개변수 갱신 방법

```
# db2 update db cfg for ‘db_name’ using ‘매개변수_이름’ ‘값’

<예>
# db2 update db cfg for sample using logbufsiz 32
```

▷ 변경된 매개변수 값은 데이터베이스가 다시 activate될 때 유효해 진다. 즉 모든 사용자의 연결이 종료된 후 다시 연결되는 경우에 변경된 매개변수 값이 적용됨

▷ Database 구성 파일 매개변수 (예)

```
데이타베이스 sample에 대한 데이타베이스 구성

 데이타베이스 구성 릴리스 레벨                  = 0x0800
 데이타베이스 릴리스 레벨                        = 0x0800
 데이타베이스 지역(territory)                    = ko_KR
 데이타베이스 코드 페이지                        = 970
```

```
 데이타베이스 코드 세트                                  = IBM-eucKR
 데이타베이스 국가 코드                                  = 82
 디렉토리 오브젝트명                          (DIR_OBJ_NAME) =
 이 데이타베이스에 대한 발견 지원            (DISCOVER_DB) = ENABLE

 병렬화 정도                                    (DFT_DEGREE) = 1
 생략시 조회 최적화 클래스                      (DFT_QUERYOPT) = 5
 산술연산 예외시 계속                        (DFT_SQLMATHWARN) = NO
 보유된 자주 사용되는 값의 수                (NUM_FREQVALUES) = 10
 보유된 quantile의 수                        (NUM_QUANTILES) = 20
 백업 보류                                               = NO
 데이타베이스가 일관성이 있음                            = YES
 롤 포워드(rollforward) 보류                             = NO
 복원 보류중                                             = NO
 다중 페이지 파일 할당 작동                              = NO
 복구를 위한 로그 보유 상태                              = NO
 로깅에 대한 사용자 나감(user exit) 상태                 = NO
 데이타베이스 힙(heap)(4KB)                      (DBHEAP) = 1200
 카탈로그 캐쉬 크기(4KB)                 (CATALOGCACHE_SZ) = 64
 로그 버퍼 크기(4KB)                             (LOGBUFSZ) = 8
 유틸리티 힙(heap) 크기(4KB)                 (UTIL_HEAP_SZ) = 5000
 버퍼 풀 크기(4KB)                               (BUFFPAGE) = 1000
패키지 캐쉬 크기(4KB)                        (PCKCACHESZ) = (MAXAPPLS*8)
 통계 힙(heap) 크기(4KB)                     (STAT_HEAP_SZ) = 4384
 교착 상태 점검 간격(ms)                       (DLCHKTIME) = 10000
 적용업무당 잠금 목록의 백분율                  (MAXLOCKS) = 10
 잠금 시간종료(초)                            (LOCKTIMEOUT) = -1
 변경된 페이지 임계값                      (CHNGPGS_THRESH) = 60
 비동기 페이지 정리자(cleaner)의 수       (NUM_IOCLEANERS) = 1
```

```
 I/O 서버의 수                                (NUM_IOSERVERS) = 3
 색인 정렬 플래그                                (INDEXSORT) = YES
 순차 검출 플래그                                (SEQDETECT) = YES
 생략시 프리페치 크기(4KB)                     (DFT_PREFETCH_SZ) = 32
 생략시 컨테이너의 수                                      = 1
 생략시 테이블공간 Extent 크기(4KB)            (DFT_EXTENT_SZ) = 32
 실행중인 프로그램의 최대 수                        (MAXAPPLS) = 40
 평균 실행중인 적용업무의 수                        (AVG_APPLS) = 1
 적용업무당 열린 DB 파일의 최대 수                  (MAXFILOP) = 64
 로그 파일 크기(4KB)                              (LOGFILSIZ) = 1000
 기본 로그 파일의 수                              (LOGPRIMARY) = 3
 2차 로그 파일의 수                               (LOGSECOND) = 2
 로그 파일에 대한 변경된 경로                     (NEWLOGPATH) =
 로그 파일에 대한 경로                                      = /home/db2inst1/db2inst1/
NODE0000/SQL00001/SQLOGDIR/
 다음에 사용할 로그 파일                                    =
 처음에 사용할 로그 파일                                    =
 그룹 확약 계수                                   (MINCOMMIT) = 1
 소프트 점검점 전에 수정된 로그 파일의 백분율  (SOFTMAX) = 100
 복구를 위한 로그 보유 작동                       (LOGRETAIN) = OFF
 로깅에 대한 사용자 나감(user exit) 작동          (USEREXIT) = OFF
 자동 재시작 작동                                (AUTORESTART) = ON
 색인 재작성 시간                                 (INDEXREC) = SYSTEM (RESTART)
 생략시 loadrec 세션의 수                      (DFT_LOADREC_SES) = 1
```

▷ Database 구성 매개변수들 중에서 자주 참조되고 사용되는 것들은 다음과 같다.

(1) DataBase Shared Memory

가. buffpage : buffer pool size

def : UNIX -> 1000 (2 * maxappls - 524288)

OS/2 -> 250 (2 * maxappls - 524288)

rem : multiple user

db server로만 machine 사용 => 환경에서 75% 사용

large data

one db on the machine

나. logbufsz : log buffer size

- out buffer : update시 disk에 log를 write하기 전에 사용
- out buffer size

def : 8 (4 - 12)

rem : 주로 read나 disk사용률(utilization)이 높으면 숫자를 올려줄 것

다. locklist : maximum storage for lock lists

- 하나의 application에 의해 사용되는 locklist의 비율이 maxlocks에 이를 때 db manager는 lock escalation에 들어간다.

def : UNIX -> 100(4 - 60000)

rem : (512 * 32(or 64) * maxappls) / 4 = 96

(2) Agent Private Memory

가. sortheap : sort heap size

def : 256(16 - 524288)

나. pakcachesz : package cache size

(host의 EDM pool과 유사)

def : 36(1 - applheapsz (128(32 - 60000))

다. maxlocks : lock의 수의 증가 전에 locklist의 최대비율

(max percent of locklist before escalation)

def : UNIX 10(1-100), OS/2 22(1 - 100)

rem : 100 * (512 locks per appl * 32 bytes * 2) / (locklist * 4096 byte)

(3) I/O & Storage

가. chngpgs_thresh : changed pages threshold

- buffer가 몇 % 차면 buffer에서 disk로 write할 것인가

  def : 60 (5 - 80)

나. num_iocleaners : number of asynchronous page cleaners

- buffer에서 disk로 write하는 역할 (마치 buffer manager처럼)

  def : 1 (0 - 255)

  rem : select only면 0도 가능 (high update면 숫자를 늘린다.)

라. num_ioservers : number of I/O servers

- prefetch I/O, backup 이나 restore utility등에 사용되는 asynchronous I/O에 사용됨

마. seqdetect : sequential detection flag

- sequential detection

  def : yes [yes, no]

(4) Application

가. maxappls : max number of active applications

- concurrent applications connected (both local and remote) to a db
- locklist, maxlocks와 관련

  def : UNIX 40(1 - 5000)

나. arg_appls : average number of active applications

- sql optimizer가 access plan결정할 때 buffer pool의 availability를 인지하게 됨

- large volumn일수록 높여줄 것

def : 1(1 - maxappls)

(5) Logging

가. logretain : Archival logging enabling

def : off

rem : 'on'으로 되어 있으면 Archival logging 사용

나. userexit : Archival logging 과 관련한 userexit 프로그램 사용 여부

def : off

rem : 'on'으로 되면 userexit 프로그램이 실행되며,
Archival logging 모드로 전환됨

다. logfilsiz : 로그파일의 크기를 4K byte 단위로 지정

def : 1,000

라.logprimary : Circular logging에서 primary 로그 파일의 수를 지정

def : 3

마.logsecond : Circular logging에서 secondary 로그 파일의 수를 지정

def : 2

바.newlogpath : 로그 파일에 대한 새로운 디렉토리를 지정

def : home/db2inst1/db2inst1/NODE0000/SQL00001/SQLOGDIR/

rem : 로그 디렉토리는 로그 파일이 존재하는 디렉토리이며 운영중에는 데이터 파일이 존재하는 디렉토리와 물리적으로 다른 디스크를 할당하는 것이 바람직하다.

4.4 스냅샵 모니터링

스냅샷 모니터링은 특정 시점에서의 데이터베이스 활동에 관한 정보를 제공한다. DB2 활동의 현재 상태에 대한 그림이다. 스냅샷을 취했을 때 사용자에게 리턴되는 정보의 양은 사용하는 모니터 스위치에 의해 결정된다. 이 스위치들은 인스턴스 또는 응용프로그램 레벨에서 설정할 수 있다.

▶ Snapshop Monitor

```
$ db2  " get  monitor  switches"
$ db2  "update  monitor  switches  using  bufferpool  on
  uow  on  "
$ db2  " get  snapshot for  all  on  db명  " | more
$ db2  " get  snapshot for  locks on  db명  " | more
$ db2  " get  snapshot for  application agentid #  " |more
         ( # 은 application handle ID임)
```

스냅샷 모니터에 사용되는 모니터링 레벨은 다음과 같다.

- 데이터베이스 관리자 – 활동중인 인스턴스에 대한 정보 수집
- 데이터베이스 – 데이터베이스 정보 수집
- 응용프로그램 – 응용프로그램 정보 수집
- 버퍼 풀 – 버퍼 풀 활동 정보 수집
- 테이블 공간 – 데이터베이스내의 테이블 공간 정보 수집
- 테이블 – 데이터베이스내의 테이블 정보 수집
- 잠금 – 데이터베이스에 대해 응용프로그램에 의해 걸린 잠금 정보 수집
- 동적 SQL – 데이터베이스에 대한 SQL 문장 캐쉬로부터 특정시점의 문장 정보 수집

4.5 Event Monitor

이벤트 모니터링은 DB2 이벤트의 특정 사건의 발생을 기록한다. 이것은 교착 상태, 연결, SQL문장들을 포함한 일시적인 이벤트에 대한 정보를 수집하도록 한다.

▶ Event Monitor

```
$ db2 create  event  monitor  testmon  for  statements
  write  to  file  --------- -------------
 (event monitor 이름) (database,transactions, tables,...)
 '/home/rdb/int1hqd1/SQL00001/db2event'  maxfiles 24
  maxfilesize  1024  nonblocked  append  "
$ db2  "set  event  monitor  testmon  state  1"
$ db2  "select  evmonname , event_mon_state(evmonname)
       from syscat.eventmonitors  "
$ db2evmon  -db  sample  -evm  testmon
$ db2evmon  -path  /home/rdb/int1hqd1/SQL00001/db2event
   (이전 작업의 대체용)
```

모니터 스위치는 다음중의 하나가 발생될 때 초기화되거나 재설정된다.

- 응용프로그램 레벨 모니터링 - 응용프로그램이 데이터베이스에 연결될 때
- 데이터베이스 레벨 모니터링 - 첫번째 응용프로그램이 연결될 때
- 테이블 레벨 모니터링 - 테이블이 첫번째 액세스될 때
- 테이블 공간 레벨 모니터링 - 테이블 공간이 처음 액세스될 때
- RESET MONITOR 명령어가 발행될 때
- 특정 모니터 스위치가 On될때

4.6 Snap Shot 내용 분석

4.6.1 Buffer Pool Size (buffpage)

▷ Buffer pool hit ratio는 page request를 위해 DBMS가 disk로부터 physical i/o를 하지 않은 횟수의%를 나타내는 것으로, hit ratio가 높을수록 disk i/o가 적었음을 의미한다.

▷ Buffer pool hit ratio 산출공식

1 - ((buffer pool data physical reads + Buffer pool index physical reads) / (buffer pool data logical reads + Buffer pool index logical reads))

⇨ hit ratio가 적으면 (close to zero), buffer pool page 개수를 늘려 준다.

▷ buffer pool data write 혹은 buffer pool index write가 buffer pool data physical read 혹은 buffer pool index physical read와 비교할 때 percentage가 높으면 buffer pool page개수를 늘려 준다.

▷ 1차적으로 database내에서 buffer pool size를 check하고 application 별로도 특정 application이 Buffer pool hit ratio가 유난히 낮은가도 check한다.

▷ total buffer pool physical read time 혹은 total buffer pool physical write time의 소요시간이 크다면 i/o wait를 수반할 수도 있으므로 data를 서로 다른 device에 두는 것도 고려할만 하다.

▷ maximun data file open 개수는 maxfilop configuration parameter에 조정된다.

▷ async physical data page read는 db manager prefetch를 통해 sync physical data page read는 db manager agent에 의해 일어난다. sync i/o가 일어나는 동안 application은 대기하게 된다.

▷ sync read에 대한 async read의 ratio로 prefetch가 얼마나 잘 작동하는가를 알 수 있고, 이것을 또한 num_ioservers configuration parameter의 기초가 된다.

가. buffer pool data logical reads

- 정의 : buffer pool을 통과하는 data page의 logical read requests 횟수
- buffer pool에 존재하던 page나 buffer pool로 읽어올 data page를 모두 포함

나. buffer pool data physical reads

- 정의 : buffer pool로 data page를 갖고오기 위하여 physical i/o를 수반한 read

requset 횟수

다. buffer pool data write

- 정의 : buffer pool data page가 physical하게 disk에 write 된 횟수를 나타냄
- buffer pool data page가 write되는 시점은
  -> 다른 data page를 read하기 위해 page를 free시킬 때
  -> buffer pool을 flush시킬 때

라. buffer pool index logical reads

- 정의 : buffer pool data logical reads와 유사 (index만 해당함)

마. buffer pool index physical reads

- 정의 : buffer pool data physical reads와 유사 (index만 해당함)

바. buffer pool index writes

- 정의 : buffer pool data writes와 유사 (index만 해당함)

사. total buffer pool physical read time

- 정의 : data 혹은 index page가 disk로부터 buffer pool로 physical i/o가 일어난 read request에 소요된 시간

아. total buffer pool physical write time

- 정의 : data 혹은 index page가 disk로부터 physical하게 write하는데 소요된 시간

자. database files closed

- 정의 : database file이 close되는 전체 횟수

차. buffer pool asynchronous data reads

- 정의 : buffer pool로 async하게 읽혀진 page갯수
- synchronous physical read 개수 =
  (buffer pool data physical reads - buffer pool async datal reads)

카. buffer pool asynchronous read time

- 정의 : db manager prefetcher에 의한 reading에 전체 소요된 시간
- synchronous reading total elapsed time = total buffer physical read time - buffer pool asynchronous read time

4.6.2 Sort (sortheap, sortheap threshold)

▷ total sort heap allocated 전체 page수로 sheapthres configuration parameter를 조정한다.

▷ post threshold sorts의 숫자가 높으면

-> sortheap threshold를 높이거나 (혹은 sortheap size를 늘려준다.)

-> sort가 적게 일어나도록 application을 조정한다.

▷ piped sort가 reject되면

-> (1) sortheap을 줄이거나

-> (2) sort 도메 threshold를 늘리는 것을 고려할 수 있다.

(2)항의 경우에는 추가allocate로 증가되는 memory로 인해 paging이 발생할 수 있고

(1)항의 경우에는 추가 merge phase를 필요로 하므로 sort의 속도를 저하시킬 수 있다.

▷ sort overflow의 빈도가 높으면 sortheap size를 늘려준다.

가. total sort heap allocated

- 정의 : db manager나 db 차원에서 어떤 시점에 Sort를 위해 sort heap space중에서 allocate되는 전체 page 개수

나. post threshold sorts

- 정의 : sort heap threshold에 도달한 후에 추가 heap을 request한 sort 횟수
- db마다 sortheap에 allocated 된 memory양이 sort heap threshold에 도달하면 db manager는 sort heap parameter value 보다는 적게 추가 sort heap을 allocate한다.

다. piped sorts requested

- 정의 : piped sorts를 request한 횟수

-> piped sort는 disk i/o를 줄이고 전체적인 system performance를 증대시킨다.

-> piped sorts는 sheapthres에 초과하지 않으면서 추가로 sort heap을 allocate할 수 있으면 sort initiate시 일어난다.

(piped sorts를 위해서는 sort heap을 적게하고, sort heap threshold를 크게한다.)

-> sql explain output을 보면 optimizer가 piped sort를 request 했는지 알 수 있다.

라. piped sorts accepted

- 정의 : piped sorts accept 된 횟수

마. total sorts

- 정의 : sort가 수행된 전체횟수

바. total sort time

- 정의 : 모든 sort가 실행되는데 소요된 전체시간 (단위 : millisec)

사. sort overflows

- 정의 : sort heap size가 부족하여 temporary storage를 위해 추가로 disk space를 요청했던 sort의 횟수

아. active sorts

- 정의 : db내에서 current하게 sortheap에 allocate된 sort 개수

-> total sort heap allocate와 비교하여 각각의 sort를 위하여 소요되는 평균 sort heap space를 계산할 수 있다.

4.6.3 Lock

▷ total lock list memoty in use를 참고하여 lock list configuration parameter를 조정한다.

▷ lock escalation을 막기위해서는

-> lock list configuration parameter를 증가시킨다.

-> maxlocks configuration parameter를 증가시킨다.

-> lock이 과다하게 일어나는 application을 파악한다.

=> maximum number of locks held 참조

▷ number of lock timeout을 참고로 locktimeout db configuration parameter조정

▷ maximum number of locks held를 기초로 maxlocks configuration parameter를 조정

가. lock held

- 정의 : current 하게 lock이 걸린 횟수

나. lock waits

- 정의 : application이나 connection에서 lock을 대기하던 전체횟수
(time waited on locks value와 함께 각각의 lock에 소요되는 평균 wait time 산정가능)

다. time waited on locks

- 정의 : lock대기에 소요된 전체시간

라. total lock list memoty in use

- 정의 : 현재 사용중인 lock list에 소요된 정체 memory (단위 : bytes)

마. deadlocks detected

- 정의 : deadlock이 발생한 전체횟수

바. lock escalations

- 정의 : row level lock에서 table level lock으로 lock이 escalate된 횟수

-> lock escalation은 하나의 application에서 걸린 전체 lock의 갯수가 lock list space의 maximum값에 도달했을 경우
(하나의 application에서 사용된 lock list의 % 값이 maxlocks에 도달했을 경우)

사. number of lock timeouts

- 정의 : timeout 된 lock의 횟수

아. maximum number of locks held

- 정의 : 한 tx에서 걸린 최대 lock의 개수

▶ SnapShot 자료 (Agent-ID SnapShot 자료)

=> db2 get snapshot for application agentid 26818

【 적용업무 스냅샷 】

에이전트 ID	= 26818
적용업무 상태	= 연결완료
상태 변경 시간	= 수집되지 않음
적용업무별로 사용되는 코드 페이지의 ID	= 970
데이터베이스의 국가 코드	= *LOCAL.db2inst1.961224004005
적용업무명	= brcm10
적용업무 ID	= *LOCAL.db2inst1.961224004005
순차번호	= 0001
권한 ID	= RDBBA
실행 ID	= root
클라이언트의 구성 NNAME	=
클라이언트 db 관리 프로그램 제품 ID	= SQL02011
클라이언트 적용 업무의 프로세스 ID	= 33873
클라이언트 적용업무의 플랫폼	= AIX
클라이언트 통신 프로토콜	= 국지 클라이언트
데이터베이스명	= LNID6DB0
데이터베이수 경로	= /home/db2inst1/SQL00003/
클리이언트 데이터베이스 별명	= LNID6DB0

연결된 이후의 잠금 시간종료의 수	= 0
잠금시 UOW가 대기하는 총 시간(밀리초)	= 수집되지 않음
잠금을 보유한 에이전트 ID	=
잠금을 보유한 적용업무 ID	=
잠금을 보유한 순차번호	=
잠금을 보유한 테이블 공간명	=
잠금을 보유한 테이블의 스카마	=
잠금을 보유한 테이블의 이름	=
잠금 모드	=
잠금 오브젝트 명	=
잠금 대기 시작 시간소인	=
총 정렬	= 0
총 정렬 시간(밀리초)	= 0
총 정렬 오버플로우	= 0
버퍼 풀 자료 논리적 읽기	= 0
버퍼 풀 자료 물리적 읽기	= 0
버퍼 풀 자료 쓰기	= 0
버퍼 풀 색인 논리적 읽기	= 0
버퍼 풀 색인 물리적 읽기	= 0
버퍼 풀 색인 쓰기	= 0

확약 명령문	= 0
구간 복원(rollback) 명령문	= 0
시도된 동적 SQL문	= 0
시도된 정적 SQL문	= 0
실패한 명령문 조작	= 0
실행된 선택 SQL문	= 0
실행된 갱신/삽입/삭제 명령문	= 0
실행된 DDL 명령문	= 0
내부 자동 리바인드	= 0
삭제된 내부 행	= 0
삽입된 내부 행	= 0
갱신된 내부 행	= 0
내부 확약	= 1
내부 구간 복원(rollback)	= 0
교착 상태로 인한 내부 구간 복원(rollback)	= 0
삭제된 행	= 0
삽입된 행	= 0
갱신된 행	= 0
선택된 행	= 0
읽은 행	= 4
기록된 행	= 0
시도된 바인드/사전 처리 컴파일러	= 0
사용된 UOW 로그 공간 (바이트)	= 수집되지 않음
이전의 UOW 완료 시간소인	= 수집되지 않음

현재 통신 힙(heap) 크기(바이트)	= 0
최대 통신 힙(heap) 크기(바이트)	= 0
연결 요청 시작 시간소인	= 06/24/1996 09:40:05.267537
연결 요청 완료 시간소인	= 06/24/1996 09:40:05.289663
최종 재설정 시간소인	=
스냅샷 시간소인	= 06/24/1996 13:16:29.320610
명령문 유형	=
명령문 조작	=
섹션 번호	=
적용업무 작성자	=
패키지명	=
커서명	=
명령문 정렬(sort)	= 0
명령문 조작 시작 시간소인	=
명령문 조작 중단 시간소인	=
명령문이 사용하는 총 사용자 CPU 시간 = 0.000000	
명령문이 사용하는 총 사용자 CPU 시간 = 0.000000	
에이전트가 사용하는 총 사용자 CPU 시간	= 0.000000

▶ SnapShot 자료 (DataBase SnapShot 자료)

=> db2 get snapshot for database on LNID6DB0

【 데이터베이스 스냅샷 】

데이터베이스명 = LNID6DB0
데이터베이스 경로 =
/home/db2inst1/db2inst1/SQL00003/
입력 데이터베이스 별명 = LNID6DB0
데이터베이스 상태 = 사용중
현재 보유된 잠금 = 0
잠금 대기 수 = 0
잠금시 데이터베이스 대기 시간(밀리초) = 0
사용 중인 잠금 일람표 메모리(바이트) = 2052
검출된 교착상태 = 0
잠금 레벨 자동 업그레이드(escalation) = 0
독점 잠금 레벨 자동 업그레이드(escalation) = 0
잠금시 대기하는 현재 적용업무 = 0
잠금 시간종료 = 0

할당된 총 정렬 힙(heap) = 0
총 정렬(sort) = 8
총 정렬(sort) 시간(밀리초) = 361
정렬(sort) 오버플로우 = 3

비동기 풀 색인 페이지 쓰기	= 0
총 버퍼 풀 읽기 시간(밀리초)	= 104
총 버퍼 풀 쓰기 시간(밀리초)	= 6
LSN 간격 정리자(Cleaner) 트리거	= 0
Dirty page steal 정리자(Cleaner) 트리거	= 0
Dirty page 임계값 정리자(Cleaner) 트리거	= 0
직접 읽기	= 2
직접 쓰기	= 4
직접 읽기 요청	= 1
직접 쓰기 요청	= 2
직접 읽기 소요 시간(밀리초)	= 17
직접 쓰기 소요 시간(밀리초)	= 36
닫힌 데이터베이스 파일	= 0
시도된 확약 명령문	= 118
시도된 구간 복원(roll back) 명령문	= 10
시도된 동적 명령문	= 628
시도된 정적 명령문	= 1477
실패한 명령문 조작	= 0
실행된 선택 SQL문	= 105
실행된 갱신/삽입/삭제 명령문	= 118
실행된 DDL 명령문	= 6
내부 자동 리바인드	= 1
삭제된 내부 행	= 0
삽입된 내부 행	= 0

첫 번째 데이터베이스 연결 시간소인	= 06/24/1996 09:39:52.659789
최종 재설정 시간소인	=
비동기식 릭기의 총 소요시간	= 18
비동기식 쓰기의 총 소요시간	= 0
비동기 읽기 요청	= 1
최종 백업 시간소인	= 05/09/1996 17:10:44.661623
스냅샷 시간소인	= 06/24/1996 13:16:56.073865
연결에 할당된 최대 페이지(high water mark)	= 6
데이터베이스 힙(heap)을 위한 최대치	= 787991
적용업무 연결	= 45
현재 연결된 적용업무	= 5
현재 db 관리 프로그램에서 실행중인 적용업무	= 0
최대 사용된 2차 로그 공간(바이트)	= 0
최대 사용된 총 로그 공간(바이트)	= 2113731
현재 할당된 2차 로그	= 0
읽은 로그 페이지	= 65
기록한 로그 페이지	= 573
패키지 캐쉬 찾아보기	= 244

4.7 성능 모니터

성능 모니터는 미리 정의한 간격(기본 간격 : 30초)으로 스냅샷 정보를 보여주기 위해 사용되는 그래픽 유틸리티이다. 인스턴스, 데이터베이스, 테이블 공간, 테이블, 연결 등과 같은 DB2 오브젝트들을 모니터할 수 있다. 성능 모니터에서의 정보는 다음의 경우에 사용된다.

- 성능 문제 감지
- 최적의 성능을 위한 데이터베이스 조정
- 성능 경향 분석
- 데이터베이스 응용프로그램의 성능 분석
- 문제의 발생 방지

성능 모니터는 제어 센터 인터페이스에서 초기화된다.
윈도우 플랫폼에서는 DB2 성능 모니터는 윈도우 환경에서 DB2 활동을 모니터하는데 사용되어질 수 있도록 윈도우 성능 모니터와 밀접하게 통합되어 있다.

오브젝트가 모니터될 때, 아이콘의 색깔은 모니터의 상태를 가리키기 위해, 녹색, 황색, 적색으로 나타난다. 색깔은 설정한 임계값에 따라 문제의 심각도를 의미한다. 녹색은 모니터가 수행되고, 모든 것이 좋다는 것을 나타낸다. 황색은 모니터가 설정한 임계값에 도달하고 있다는 경고를 나타낸다. 적색은 모니터가 임계값에 도달했다는 경보를 나타낸다.

성능 모니터는 기존 문제를 모니터할 필요가 있을 때 또는 시스템의 성능을 관찰하고자 할 때 사용한다. 특정 시점에서 데이터베이스 활동과 성능 데이터의 스냅샷을 수행한다. 이 스냅샷은 시간 경과에 따른 비교시에 사용된다. 성능 그래프에서의 각 점은 데이터 값을 표현한다.

DB2에서의 성능 모니터링은 DB2가 설치될 때 제공한 미리 정의된 모니터들을 사용하여 이루어지거나, 미리 정의된 모니터들을 복사, 수정하여 수행된다. DB2와 함께 제공되는 미리 정의된 모니터들은 다음과 같다.

- Capacity – 시스템 용량에 대한 정보 수집시 사용됨.
- Sort – 정렬 힙과 정렬 힙 임계값 매개변수가 적절하게 설정된 가를 확인하기 위해, 시스템 시작시, 활동 절정기 때, 응용프로그램이 변경될 때 수행함.

- Locking – 잠금이 시스템에 얼마나 발생되는가와 잠금 목록 매개변수가 적절하게 설정되는지를 결정하기 위해 사용됨.
- Cache – 캐쉬 사용을 최적화하기 위해 사용됨.
- Deadlocks – 응용프로그램이 교착상태에 있는지를 결정하기 위해 사용됨.
- Prefetchers – 시스템에 정의된 프리페처가 충분한지를 결정하기 위해 사용됨.
- Disk Performance – 데이터베이스와 테이블 공간 레벨에서 디스크 성능에 촛점을 맞추어 성능 변수를 모니터링함.
- Global Memory – 응용프로그램 메모리 사용을 보기위해 사용됨.
- Long Running Query – 쿼리가 완료하기 위해 오랜 시간이 걸리는 원인을 파악하는데 사용됨.

이용 가능한 모니터의 목록을 보기위해서는 제어 센터에서 시스템 폴더에 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 모니터 목록을 선택한다. 모니터 목록 창은 다음과 같다.

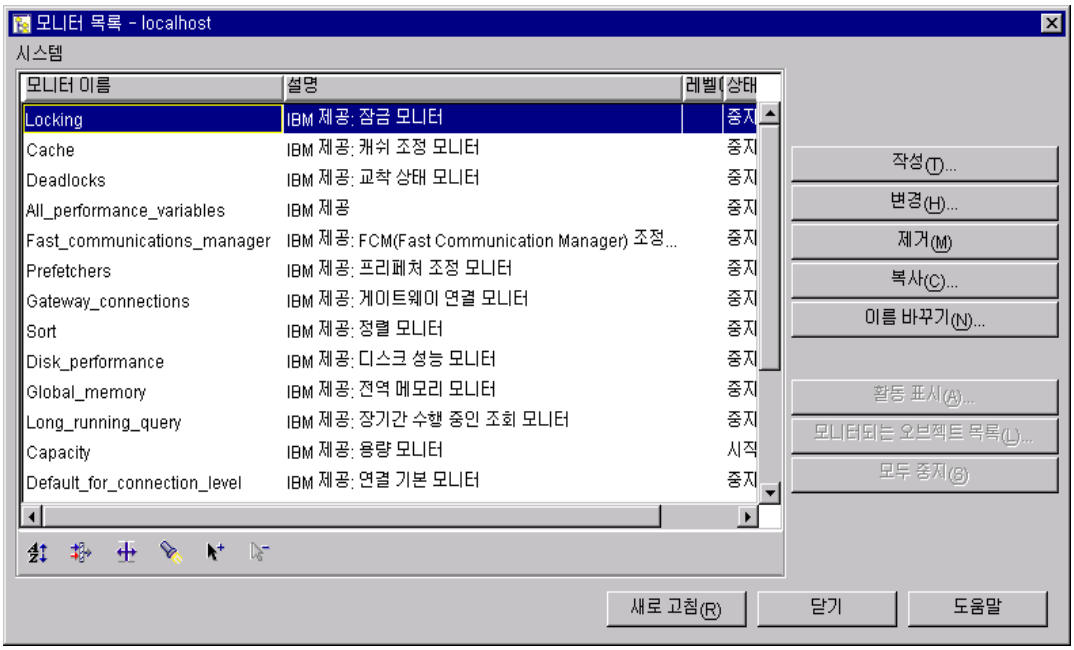


그림 4-7-1. 성능 모니터 목록 창

그림 4-7-1은 미리 정의된 모니터들을 보여준다. 창의 오른쪽에는 모니터에 대한 다양한 작업을 수행하도록 버튼들이 포함되어 있다. 이 창에서 모니터들을 작성, 변경, 제거, 복사, 이름 바꾸기를 할 수 있다. 하지만, 미리 정의된 모니터에 대하여, 이름, 공식, 텍스트 설명을 변경할 수 없다. 미리 정의된 모니터에 대해 할 수 있는 것은 임계값을 변경하거나, 경보 활동을 변경하거나, 복사하는 것이다.

모니터 목록 창에서 모니터를 선택하고, 복사 버튼을 클릭한 다음, 복사 창에서 새로운 모니터 이름을 입력한다. 모니터 목록 창에서는 새로운 모니터 이름을 볼 수 있다. 그리고 나서 새로운 모니터를 선택하고 변경 버튼을 클릭한다. 그림 4-7-2는 모니터 변경 창을 보여준다. 예에서의 새로운 모니터는 미리 정의된 모니터중의 하나인 Default_for_database_level 모니터에서 복사하고, Database_New 모니터라는 이름으로 저장한다.

이 창에서는 모니터가 포함한 성능 변수들을 보여준다. 더 많은 성능 변수들을 이 모니터에 추가하기 원한다면, 추가 버튼을 클릭한다. 이용가능한 모든 성능 변수들이 화면에 보여지고, 그 중에서 선택할 수 있다 (그림 4-7-3).

그림 4-7-2의 하단은 그래프 설정값을 보여준다. 또한 각 성능 변수에 대한 임계값을 설정할 수 있다. 선택된 성능 변수들에 대한 경고와 경보 지대 경계들을 정의할 수 있다.

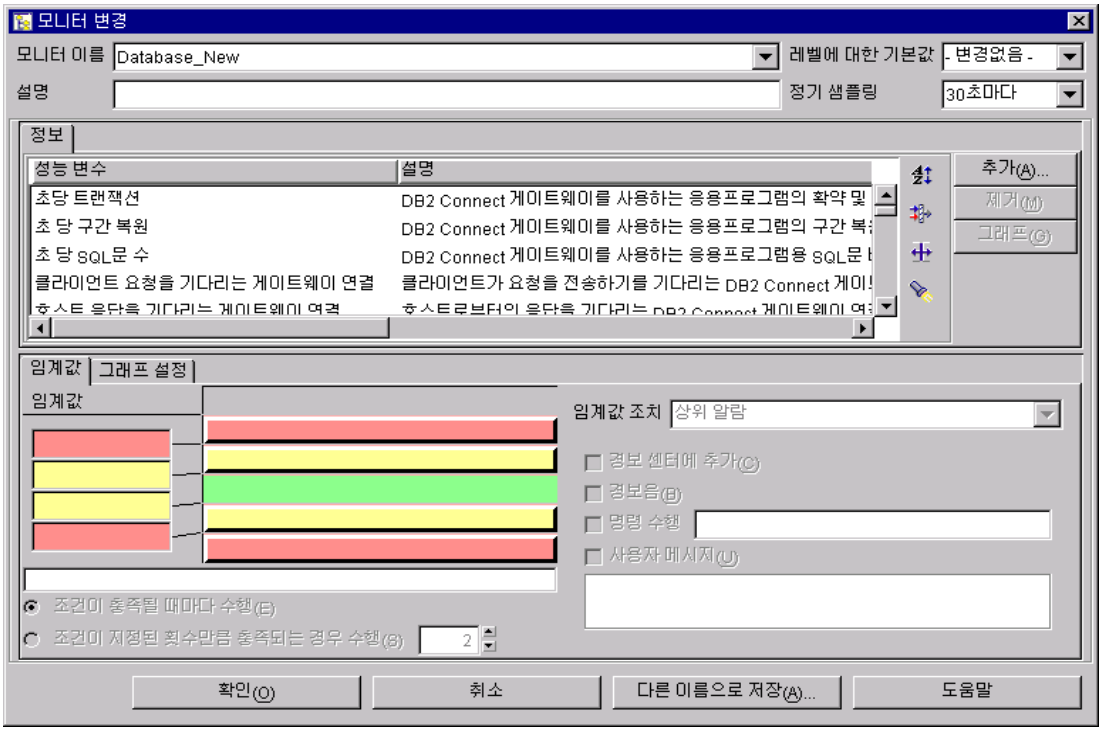


그림 4-7-2. 성능 모니터 변경 창

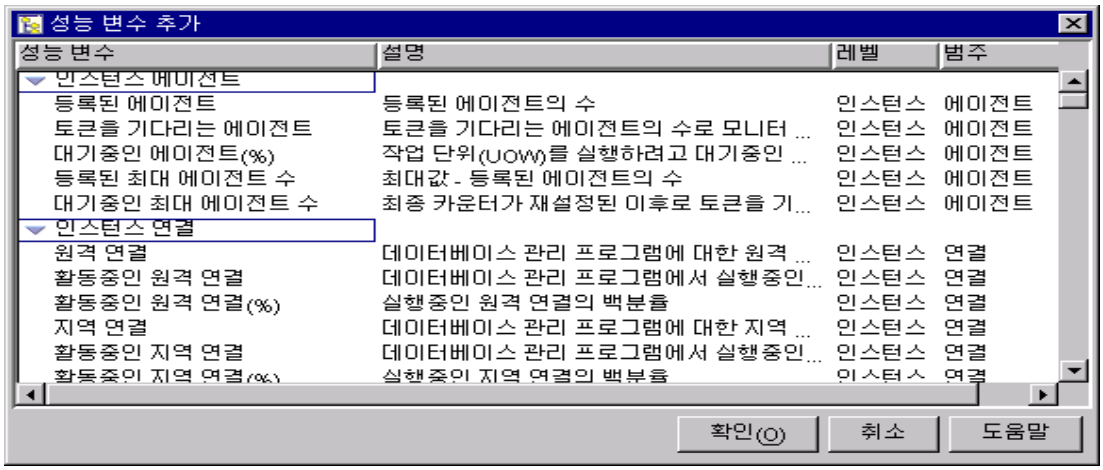


그림 4-7-3. 성능 모니터에 성능 변수 추가

임계값을 정의할 때, 값이 이 임계값을 초과할 때 DB2가 응답하는 방법을 설정할 수 있다. 가능한 응답으로는 다음과 같다.

- 엔트리를 경보 센터에 추가한다.
- 경고음을 발행한다.
- 프로그램 또는 스크립트를 시작한다.
- 팝업 메뉴를 띄움으로써 사용자에게 메시지를 보낸다.

데이터베이스 오브젝트 SAMPLE은 미리 정의된 Default_for_database_level 모니터에서 복사한 Database_New 모니터를 사용하여 모니터된다. SAMPLE 데이터베이스 아이콘에 오른쪽 클릭하고나서, 성능 모니터링과 모니터 시작을 선택한다. 그림 4-7-4은 모니터 시작 창을 보여준다. 시작하고자 하는 모니터(예에서는 Database_New)를 선택하고, 확인 버튼을 클릭한다. 제어 센터 창에서, 모니터되는 데이터베이스 오브젝트 SAMPLE 아이콘이 녹색으로 변한다.

인스턴스 레벨, 데이터베이스 레벨, 테이블 공간 레벨, 테이블 레벨, 연결 레벨에 대해 기본 모니터로써 어느 모니터를 사용할 지를 지정할 수 있다. 예를 들어, 만약 데이터베이스 레벨을 기본 모니터로 변경하고자 한다면, 제어 센터에서 데이터베이스 폴더에 오른쪽 클릭을 하고 성능 모니터링, 기본 모니터 변경을 선택한 후, 화면이 보여지는 모니터들 중에서 기본 모니터로 지정하기를 원하는 모니터를 선택한다.

성능 모니터를 시작할 때에는 그림 4-7-4에서 보듯이 시작 모니터 창에서 모니터를 선택할 수 있을 뿐만 아니라, 미리 기본 모니터를 정의하여 시작할 수 있다. 데이터베이스 레벨의 기본 모니터를 시작하기 위해서는 데이터베이스 오브젝트에 오른쪽 클릭을 한 후 성능 모니터링, 시작 기본 모니터를 선택한다.

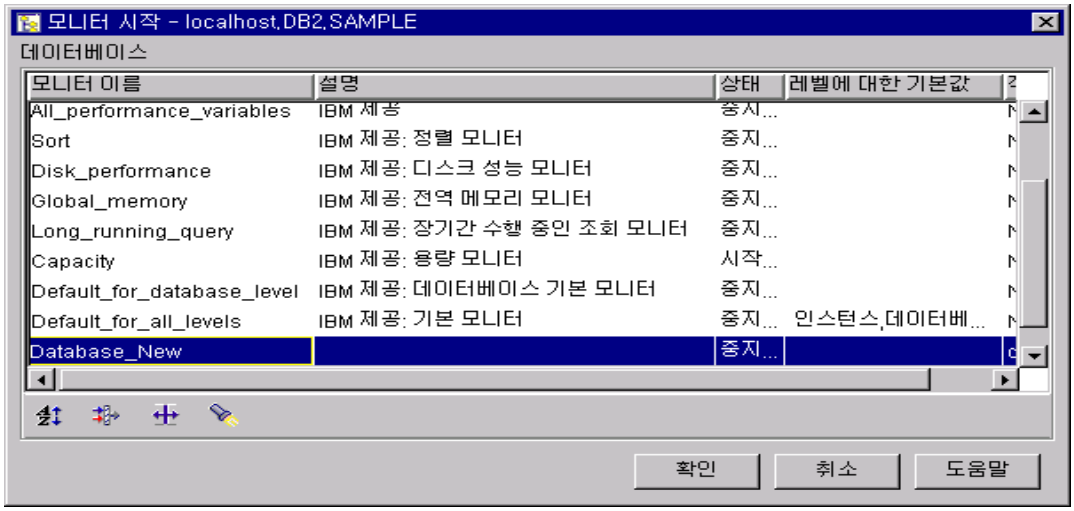


그림 4-7-4. 모니터 시작 창

모니터링이 시작되면, 성능 모니터가 수집하는 정보들이 화면에 보여진다. Database_New 모니터가 수집하는 정보들을 보기 위해서는 SAMPLE 데이터베이스 아이콘에 오른쪽 클릭을 하고, 성능 모니터링, 모니터 활동 표시를 선택한다. 그림 4-7-5는 모든 성능 변수들의 값이 모니터되는 것이 보여진다. 그림 4-7-5에서는 창의 상단은 각 성능 변수의 자세한 데이터가 보여진다. 또한 데이터베이스 요약이나 정보를 클릭하면, 성능 변수의 요약 데이터나 각 성능 변수의 설명을 볼 수 있다. 창의 하단의 그래프는 정의한 임계값을 기준으로 그려진다. 만약 성능 변수의 값이 상위 경고 값과 하위 경고 값 사이라면, 그래프는 녹색 밴드에서 그려진다.

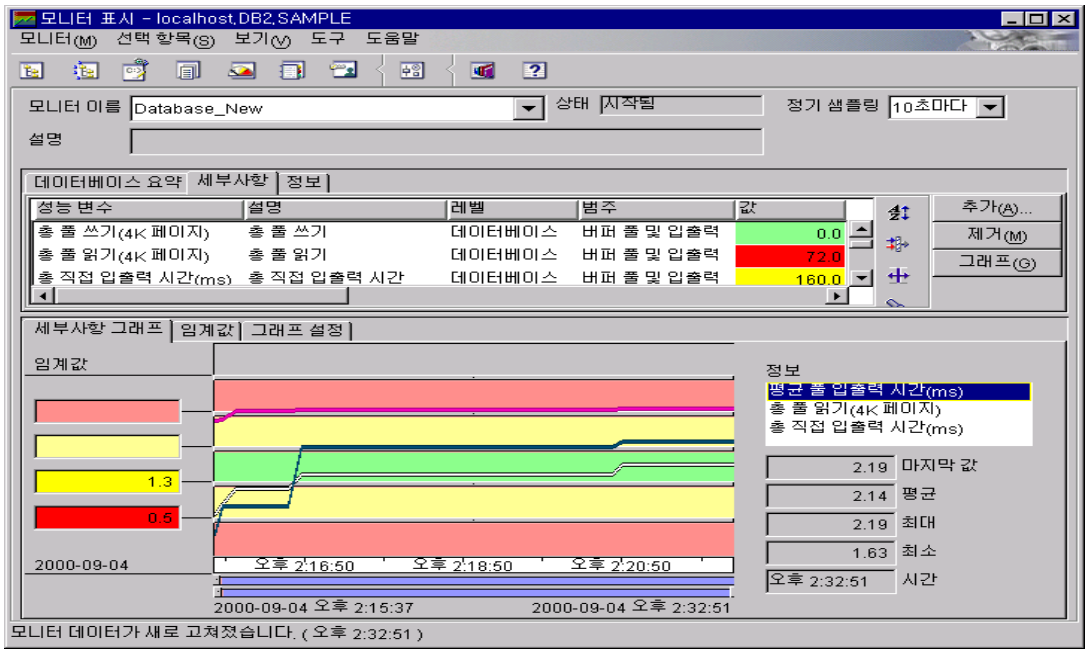


그림 4-7-5. 모니터 표시 창

4.8 성능 구성 스마트 가이드

성능 구성 마법사는 새로운 DBA이거나 가끔씩 데이터베이스를 관리하는 사람에게는 구성 매개변수들을 조정하기 위한 매우 유용한 툴이다. 경험있는 DBA라 하더라도 새로운 데이터베이스를 생성할 때 이 툴을 사용할 것을 추천한다. 즉 데이터베이스 관리자와 데이터베이스 구성 매개변수들의 기본 값 대신 권장 값을 가지고 시작할 수 있다.

성능 구성 마법사에서는 인스턴스당 하나의 데이터베이스가 필요로 하는 메모리 할당과 성능 조정을 구현하여 모든 구성 매개변수값들을 최대 성능을 위한 최적의 값으로 직접 변경하거나 변경을 위한 값들을 제시한다. 다수의 구성 매개변수들은 기본 값을 가져도 무방하지만, 데이터베이스의 최적의 성능을 보장하려면 성능 조정을 통한 해당 매개변수들의 갱신이 불가피하다. 대체로 성능 구성 마법사에서 추천되는 값들은 작업의 워크로드와 특정 서버에 대한 정보를 통한 성능 조정이므로 구성 매개변수들이 기본 값일 때보다 보다 향상된 성능을 제공한다. 그러나 이 값들은 데이터베이스 성능을 최적으로 만드는 값이라기 보다는 최적의 성능을 구현하기 위한 출발점이 되는 값들이라고 볼 수 있다.

성능 구성 마법사를 사용하는 방법은 다음과 같다.

1. 제어센터에서 성능 구성을 하고자 하는 데이터베이스를 선택한 다음 마우스의 오른쪽 버튼을 누르면 팝업 메뉴가 나타난다. 이 팝 업메뉴에서 “마법사를 사용한 성능 구성”을 선택한다. 성능 구성 마법사 소개 페이지(그림 1)에서는 구성하고자 하는 데이터베이스에 대한 정보를 보여준다.

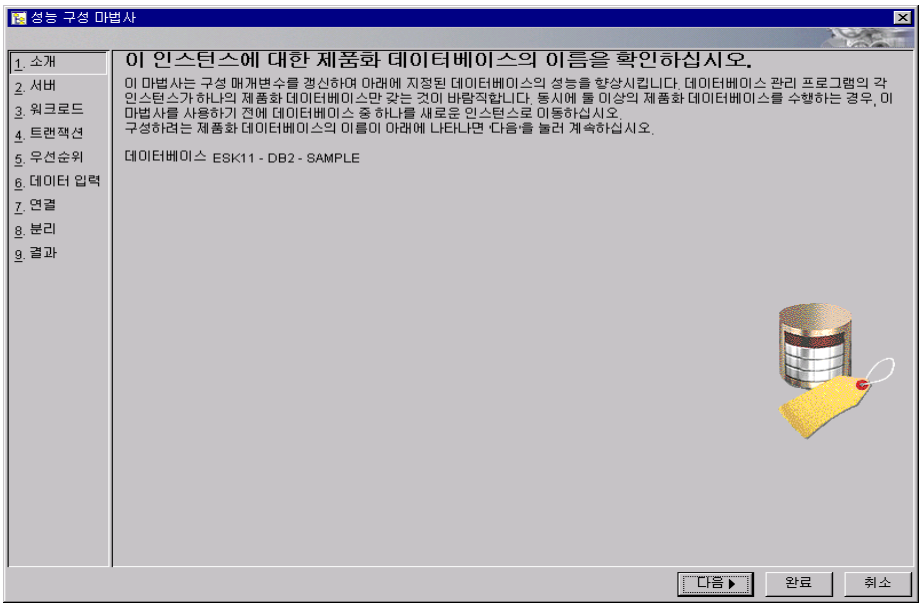


그림 1. 성능 구성 마법사의 소개 페이지

2. 각 페이지를 통해, 필요한 정보를 변경한다. 성능 구성 마법사는 제공한 각각의 값을 기반으로 데이터베이스 관리자와 데이터베이스 구성 매개변수의 값을 계산하고 적당한 값을 제공한다. 성능 구성 마법사는 7 페이지로 구성된다. 각 페이지에서 값을 입력하거나 선택한다.

   서버 페이지(그림 2)에서 데이터베이스에서 사용될(운영체제에서 사용되는 부분은 제외) 서버의 메모리(RAM)의 양을 슬라이더를 이용하여 지정한다. 만약 다른 응용 프로그램이 서버에서 수행된다면, 데이터베이스에서 사용될 서버의 메모리(RAM)의 양을 나타내는 슬라이더를 100%이하이어야 한다.

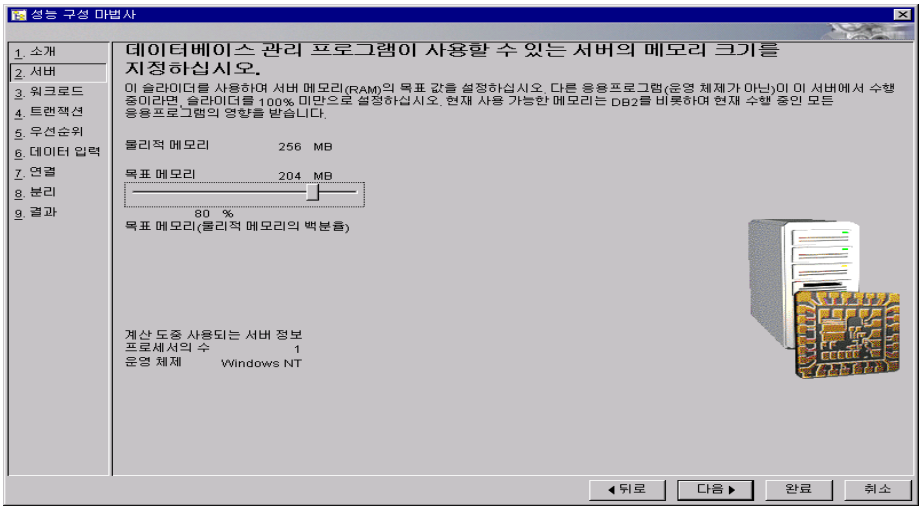


그림 2. 성능 구성 마법사의 서버 페이지

3. 워크로드 페이지(그림 3)에서 데이터베이스 워크로드 유형에 대한 옵션을 선택한다.

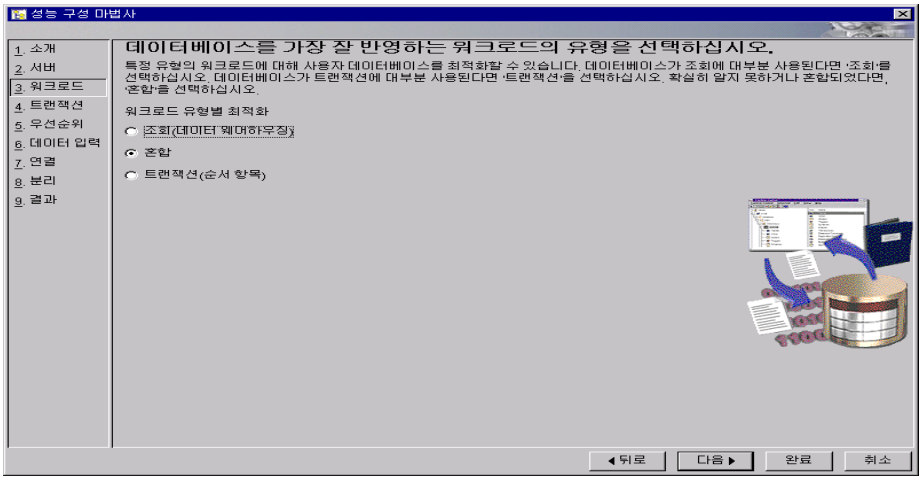


그림 3. 성능 구성 마법사의 워크로드 페이지

4. 트랜잭션 페이지(그림 4)에서 데이터베이스에 최대로 반영되는 하나의 UOW 에서의 SQL문 갯수의 근사치를 지정한다. 또한 데이터베이스내의 분당 트랜잭션의 수를 추산한다. 만약 정확한 값을 모른다면 마법사에서 제공하는 기본 값을 사용한다.

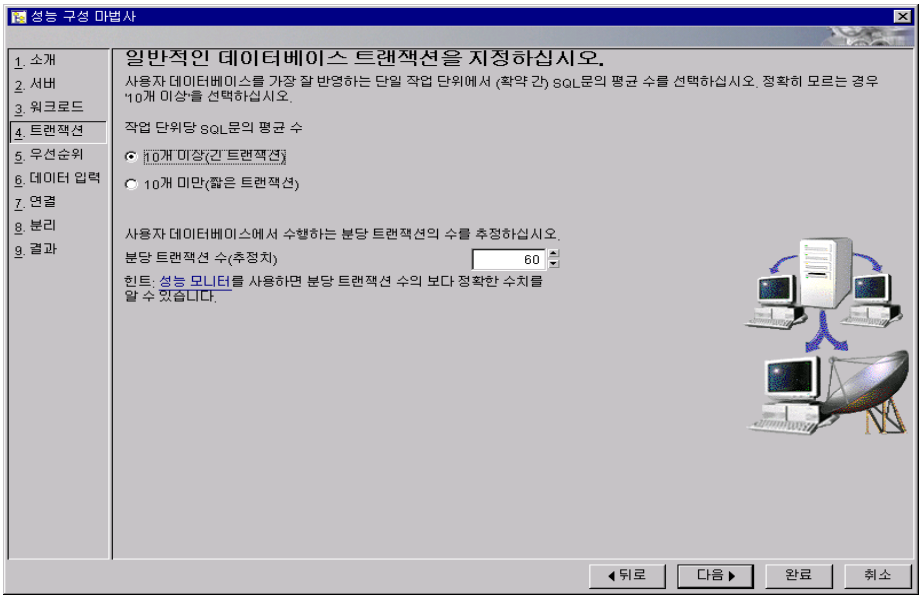


그림 4. 성능 구성 마법사의 트랜잭션 페이지

5. 우선순위 페이지(그림 5)에서 데이터베이스를 복구하는 데 필요한 시간 또는 트랜잭션 성능 중 어느 것을 최적화시키는 것이 중요한 지를 결정한다.

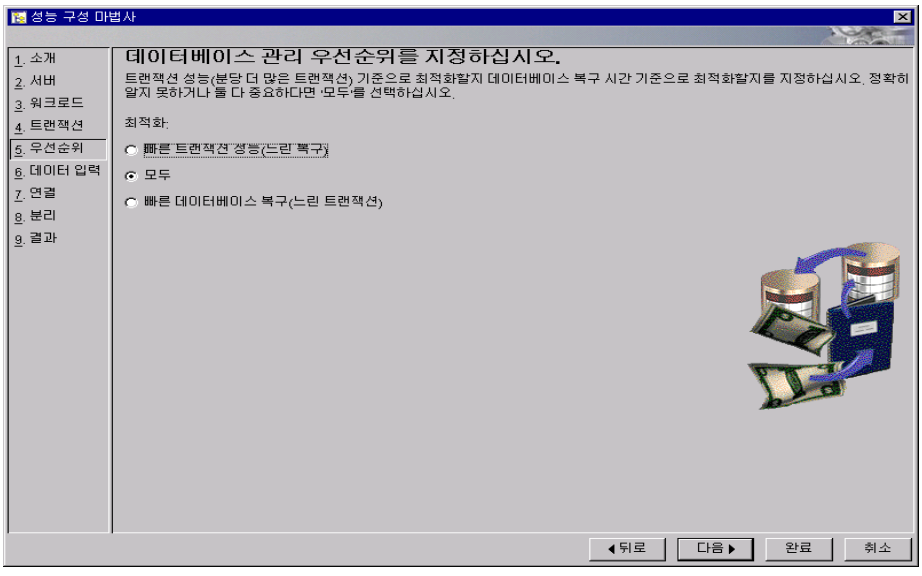


그림 5. 성능 구성 마법사의 우선순위 페이지

6. 데이터 상주화 페이지(그림 6)에서 데이터베이스는 실제 사용중인 데이터인지 아닌지를 나타낸다. 만약 해당 데이터베이스가 신규 데이터베이스이면, 데이터를 삽입한 다음 마법사를 재수행한다.

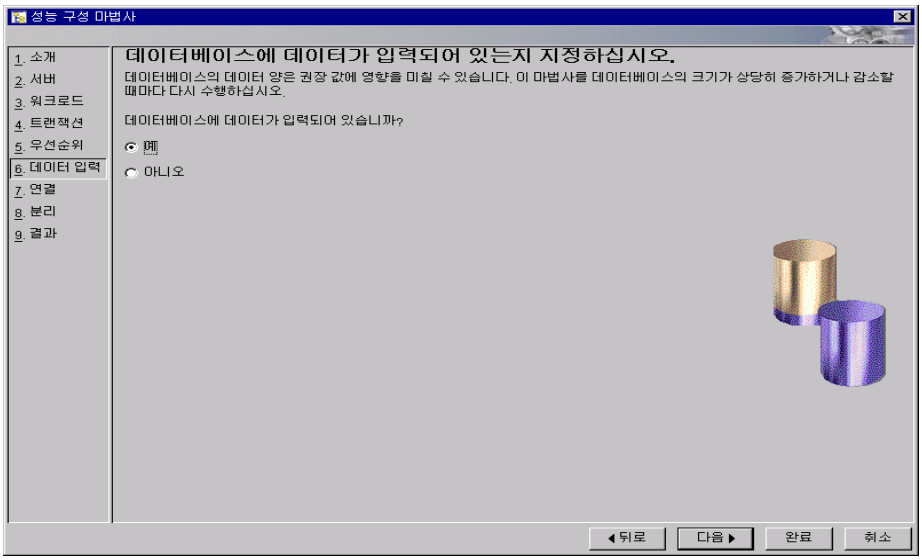


그림 6. 성능 구성 마법사의 데이터 상주화 페이지

7. 연결 페이지(그림 7)에서 해당 데이터베이스에 연결되어 사용되는 응용 프로그램의 수를 추산한다. 국지 및 원격 응용 프로그램들의 수에 대한 근사치를 지정한다.

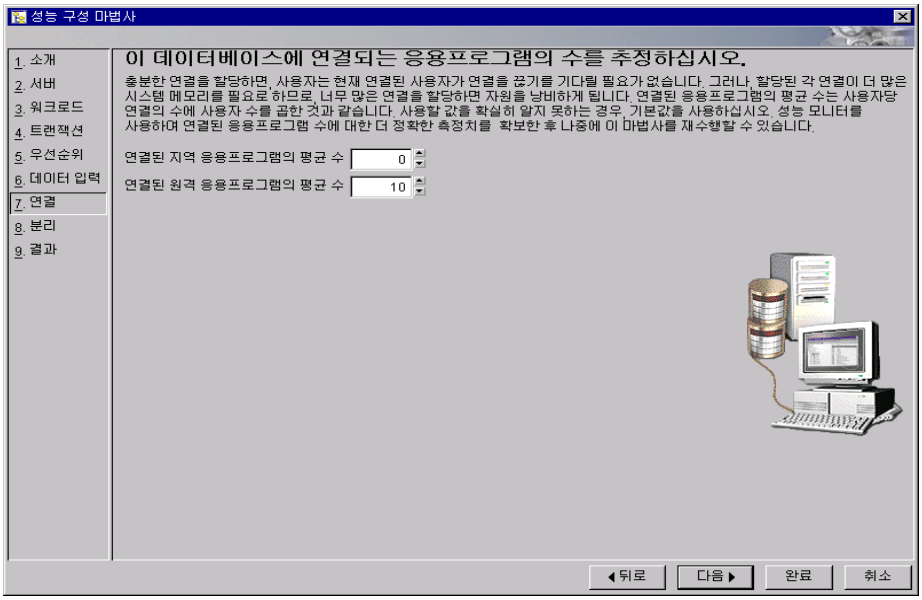


그림 7. 성능 구성 마법사의 연결 페이지

8. 분리 레벨 페이지(Isolation Level, 그림 8)에서 응용 프로그램이 최대로 반영하는 분리 레벨을 선택한다. 이러한 분리 레벨은 사용자가 데이터를 읽거나 갱신할 때 잠긴 행(Locked Row)의 수와 잠긴 기간(Lock Duration)을 결정한다. DB2는 데이터베이스를 동시 사용중인 트랜잭션들에 대해 데이터 무결성(현실세계와 데이터베이스에 입력된 데이터의 일치성)을 보장하기 위해 잠금(Locking)을 사용한다. 이 잠금(Locking)은 한 트랜잭션이 완전히 끝날 때까지 해당 데이터베이스에 대한 제어를 유지할 수 있도록 한다. 즉 진행중인 갱신 트랜잭션이 완전히 끝나기 전에 다른 응용 프로그램이 해당 행을 변경하는 것을 방지한다.

   다음의 분리 레벨들중 하나를 선택한다.
   - 만약 다수의 긴 잠금을 갖는다면, “반복가능 읽기(Repeatable Read)”를 선택한다.
   - 만약 소수의 긴 잠금을 갖는다면, “읽기 안정성(Read Stabiity)”를 선택한다.
   - 만약 다수의 짧은 잠금을 갖는다면, “커서 안정성(Cursor Stability)”를 선택한다.
   - 만약 잠금을 사용하지 않으면, “미확약 읽기(Uncommitted Read)”를 선택한다.

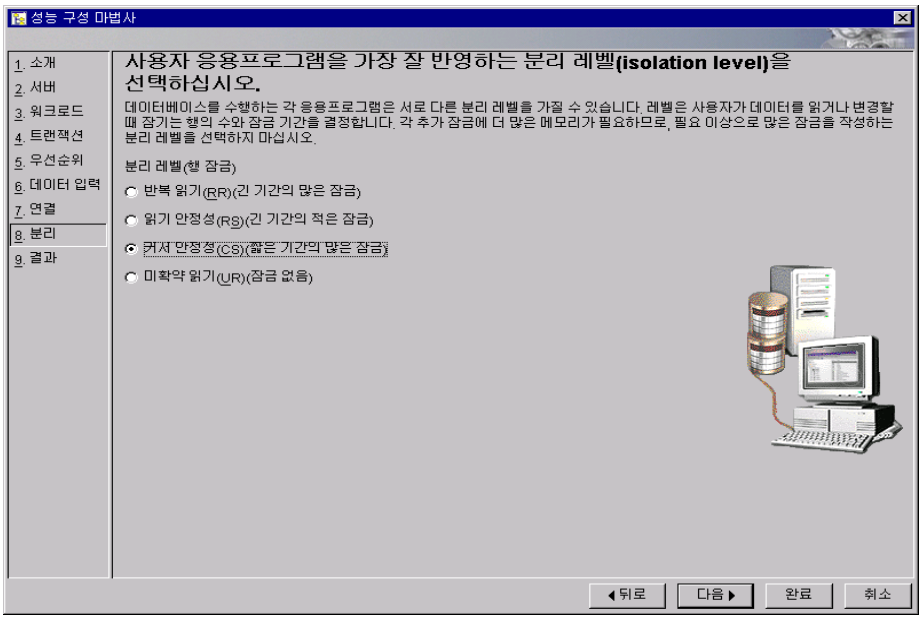


그림 8. 성능 구성 마법사의 분리레벨 페이지

9. 결과 페이지(그림 9)에서 성능구성에 대한 추천값을 검토할 수 있다. 임의의 값을 변경하려면 변경할 이전 페이지로 되돌아가야 한다. 결과 페이지에서 변경된 값을 직접 적용시키거나 나중에 스크립트 센터에서 수행할 스크립트 파일로 저장할 수 있다. 구성 매개변수에 변경될 값들이 적절하다고 생각되면 “완료” 버튼을 클릭한다. 이때 성능 구성 마법사는 데이터베이스 관리자와 데이터베이스 구성 매개변수에 적절한 값을 계산한다. 그

림 9는 각 구성 매개변수의 현재 값과 제안된 값들을 보여준다. 이 권장 값들은 즉시 적용하거나 추후 적용하기 위해 스크립트 센터로 저장할 수 있다.

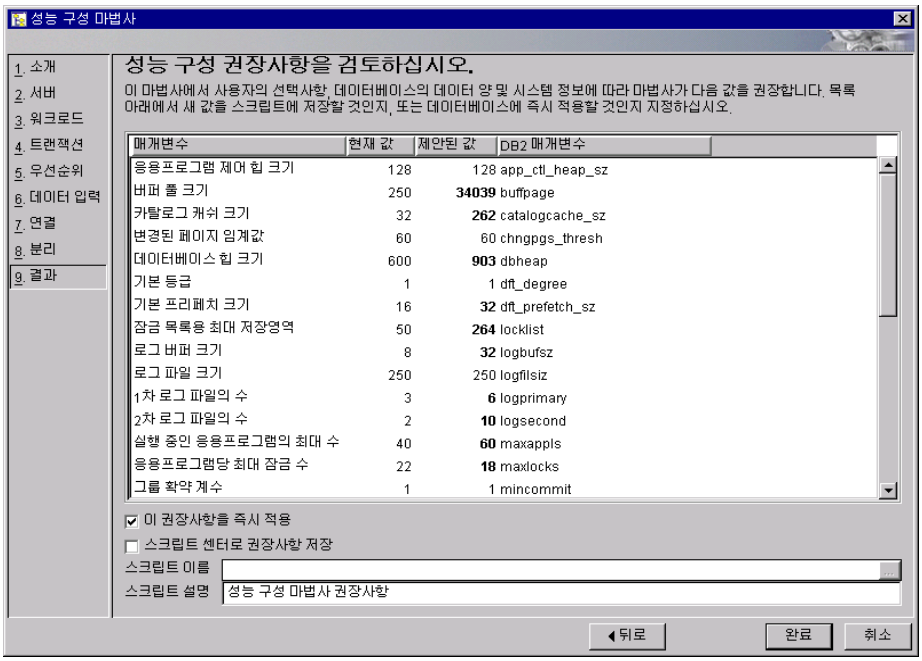


그림 9. 성능 구성 마법사의 결과 페이지

데이터베이스 구성 매개변수의 갱신을 적용시키려면, 모든 응용 프로그램을 해당 데이터베이스로부터 끊어 버리고, 그 데이터베이스로의 최초의 재연결할 때부터 구성 매개변수의 변경된 값이 적용된다. 데이터베이스 관리 프로그램 구성 매개변수(DB2 인스턴스의 구성변수)의 갱신된 값을 적용시키려면 DB2 인스턴스를 정지하고 재시작해야 한다.

데이터베이스 크기가 상당히 증가(예를 들면, 20% 이상의 데이터베이스의 크기의 증가)하거나 기계 특성이 변경(예를 들면, 더 많은 메모리의 사용이 가능)되면 성능 구성 마법사를 재수행한다. 그 이유는 이러한 데이터베이스 크기의 증가나 시스템에 메모리를 추가하는 것은 성능 변수의 값의 권장 값을 상당히 변화시키기 때문이다.

4.9 SQL Access Plan 분석 툴

▶ db2expln

▷ db2expln 툴은 시스템 카탈로그 테이블에 저장되어 있는 패키지의 정적 SQL에 대한 선택된 액세스 플랜을 기술한다.

```
# db2expln   -d   (db 명)  -c  (user명)  -p  (package명)
             -s  (section 번호) -o   (outfile 이름)
```

▷ Interactive SQL에 대해서는

```
export DYNEXPLN_OPTIONS = 'blocking all isolation ur queryopt 3'
dynexpln   <dbname>  "<SQL statement>"
```

▶ EXPLAIN

▷ EXPLAIN문은 제공된 설명 가능한 명령문에 선택된 액세스 플랜에 관한 정보를 보관하고, 이 정보를 Explain 테이블에 둔다.

▷ 설명가능 명령문은 DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES 또는 VALUES INTO SQL문이다.

▷ 이 명령문은 적용 업무 프로그램에 포함되거나 대화식으로 발행될 수 있다.

▷ 이 명령문은 동적으로 준비될 수 있는 실행 가능한 명령문이다.

ex1> 간단한 SELECT문을 설명하고 QUERYNO = 13이라는 태그를 표시한다.

```
EXPLAIN  PLAN  SET  QUERYNO = 13  FOR  SELECT  C1  FROM  T1
```

ex2> 간단한 SELECT문을 설명하고 QUERYTAG = 'TEST13' 표시를 한다.

```
EXPLAIN  PLAN  SELECTION  SET  QUERYTAG = 'TEST13'
  FOR  SELECT  C1  FROM  T1
```

ex3> 간단한 SELECT문을 설명하고 QUERYNO = 13과

QUERYTAG = 'TEST13'으로 태그를 붙인다.

```
EXPLAIN  PLAN  SELECTION  SET QUERYNO = 13 SET QUERYTAG =
  'TEST13'   FOR  SELECT  C1  FROM  T1
```

ex4> Explain 테이블이 존재하지 않으면 Explain 정보를 확보한다.

```
EXPLAIN  ALL  FOR  SELECT  C1  FROM  T1
```

▶ Visual Explain

▷ Visual Explain을 실행하기 위해 다음과 같이 explain 정보를 저장할 테이블을 생성해야 한다.

```
/$HOME/sqllib/misc> db2   connect   to   sample
/$HOME/sqllib/misc> db2   -tf   EXPLAIN.DDL
```

▷ 현재의 explain snapshot을 변경

```
db2   set   current   explain   snapshot   yes
```

▷ bindfile 내의 source 를 보기 위해 다음을 수행한다.

```
db2bfd   bindfile_name
```

Visual Explain은 데이터베이스 관리자나 응용프로그램 개발자에게 옵티마이저가 결정하는 액세스 플랜을 조사할 수 있도록 하는 GUI 유틸리티이다. Visual Explain은 스냅샷 옵션을 이용하여 생성한 액세스 플랜만을 사용할 수 있다.

Visual Explain은 이미 생성한 Explain 스냅샷을 분석하거나 Explain 데이터와 Explain 동적 SQL 문장들을 수집하기 위해 사용될 수 있다. 만약 Explain 테이블이 Visual Explain을 시작할 때 생성되어 있지 않다면, 시작시에 생성된다. Visual Explain은 명령 센터 또는 제어 센터에서 호출할 수 있다.

제어 센터 인터페이스에서, Explain 스냅샷이 저장되는 데이터베이스에 오른쪽 클릭을

한다. Explain된 명령문 실행기록 표시라고 불리우는 옵션은 Explain 데이터를 수집하고 동적 SQL 문장의 그래픽 표현을 보여준다. 이것은 단일 SQL 문장을 Explain하기 위한 가장 쉬운 방법이다.

Explain된 명령문 실행기록 표시 창이 열리면, 모든 Explain된 문장들이 표시된다. 전체 비용과 SQL 문장이 보여진다.

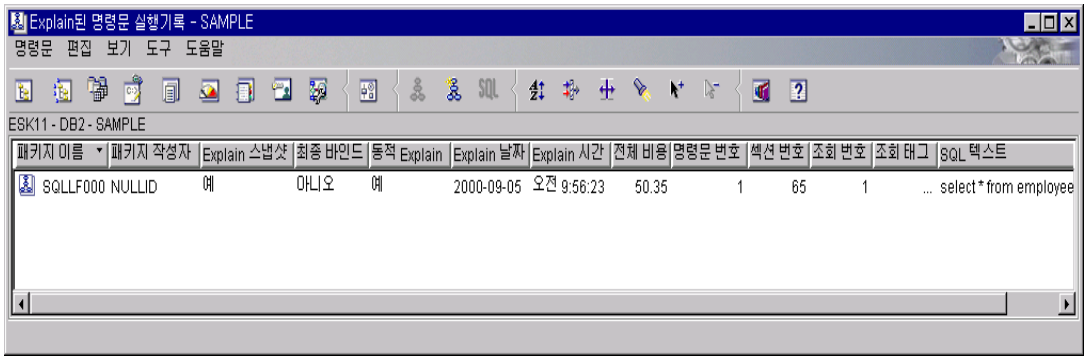


그림 1. Explain된 명령문 실행기록 창

액세스 플랜을 자세히 조사하기 위해서, Explain된 문장에 단순히 더블 클릭하거나 관심있는 항목을 선택한 후, 팬넬 메뉴에서 명령문 -> 액세스 플랜 표시를 선택한다. 모든 Expalin 문장들이 Explain된 명령문 실행기록 표시에 보여지나, EXPLAIN SNAPSHOT 정보를 가진 Explain된 문장들만 Visual Explain을 사용하여 조사할 수 있다.
Explain된 명령문 실행기록 표시 창에서 나열된 Explain 스냅샷에 주석을 추가할 수 있다. 쿼리를 설명하는 주석을 추가하기 위해서 쿼리를 선택한 후, 명령문 -> 변경을 선택한다. Explain 스냅샷은 제거하고자 하는 엔트리를 선택한 후, 명령문 -> 제거를 선택하면, Explain 테이블에서 제거될 수 있다.

Visual Explain 출력물은 SQL 문장의 구성요소를 표현하는 계층적인 그래프를 보여준다. 각 쿼리의 부분은 그래픽 오브젝트로써 표현된다. 이들 오브젝트들은 노드라 한다. 노드에는 두가지 기본 유형이 있다.

- OPERATOR 노드는 데이터의 그룹에서 수행되는 행위를 가리킨다.
- OPERAND 노드는 Operator 행위가 발생하는 데이터베이스 오브젝트를 보여준다. Operand는 Operator에 따른 오브젝트이다. 이들 데이터베이스 오브젝트들은 대개 테이블과 색인들이다.

최적의 액세스 플랜을 결정하기 위해 DB2 옵티마이저가 사용할 수 있는 Operator는

많은데, Visual Explain에 의해 사용되는 몇몇의 Operator는 다음 그림 2와 같다.

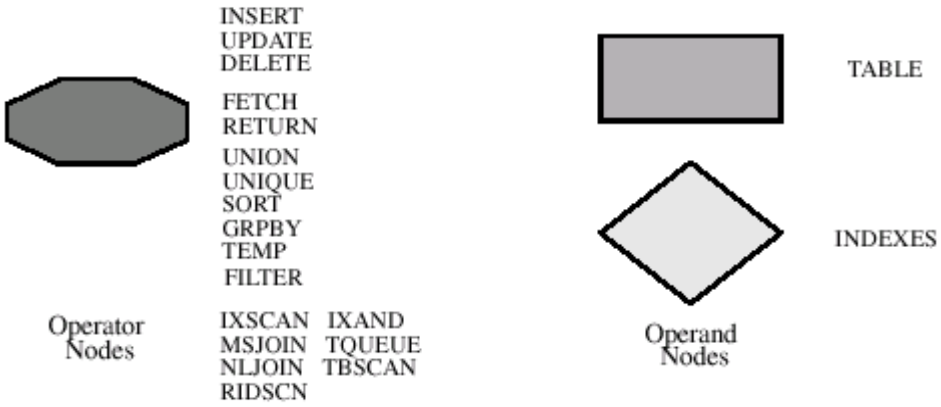


그림 2. Visual Explain에 보여지는 Operator와 Operand

이들 Operator는 데이터가 액세스하는 방법(IXSCAN, TBSCAN, RIDSCN, IXAND), 테이블이 내부적으로 조인하는 방법(MSJOIN, NLJOIN), 정렬이 필요하는지(SORT)와 같은 요소들을 가리킨다.

Visual Explain 그래픽 출력에서 보여지는 오브젝트들은 한 노드에서 다른 노드로 데이터의 흐름을 보이는 화살표에 의해 연결된다. 액세스 플랜의 마지막은 항상 RETURN Operator이다.

그림 3의 액세스 플랜은 단순한 SQL 문장이다 : SELECT * FROM DB2ADMIN.EMPLOYEE. 이 예에서는 두가지 Operator와 하나의 Operand가 있다. Operand는 DB2ADMIN.EMPLOYEE 테이블이고, Operator는 테이블 스캔(TBSCAN)과 RETURN Operator를 포함한다.
SQL 문장에 대한 Explain 데이터를 수집하는 것은 DB2 옵티마이저가 결정한 액세스 플랜을 분석하기 위한 단 하나의 방법이다. 액세스 플랜 그래프에서 보여지듯이, 각 노드는 노드에서 더블 클릭을 하거나 노드 메뉴 항목에서 세부사항 표시 옵션을 선택하면, 자세한 정보를 볼 수 있다.

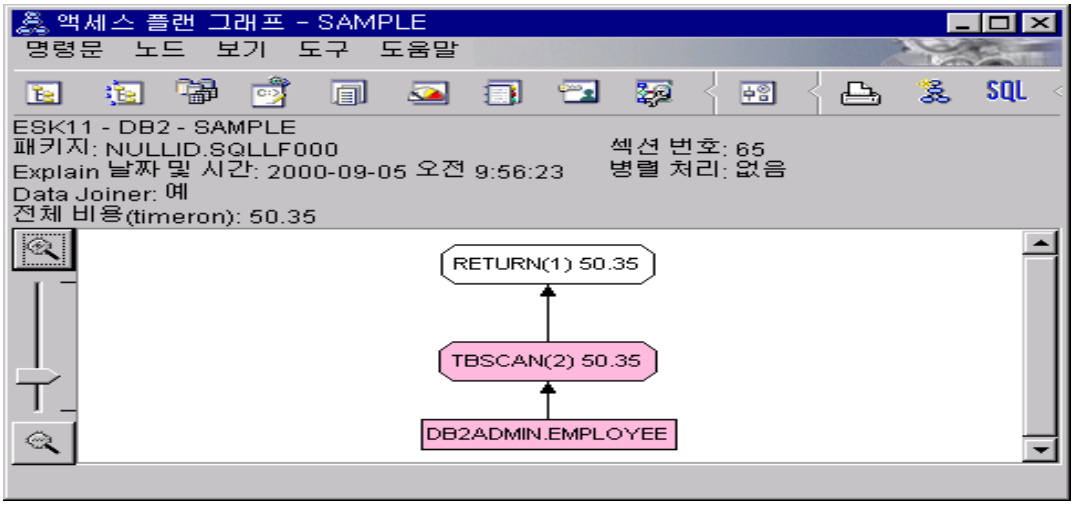


그림 3. Visual Explain : SQL 문장에 대한 그래픽 액세스 플랜

테이블 스캔 운영의 자세한 내용을 보기 위해서, TBSCAN Operator 노드를 선택한 후, 노드 메뉴 항목에서 세부사항 표시를 선택한다. 액세스 플랜에서 TBSCAN 운영에 대한 정보는 다음 그림 4와 같다.

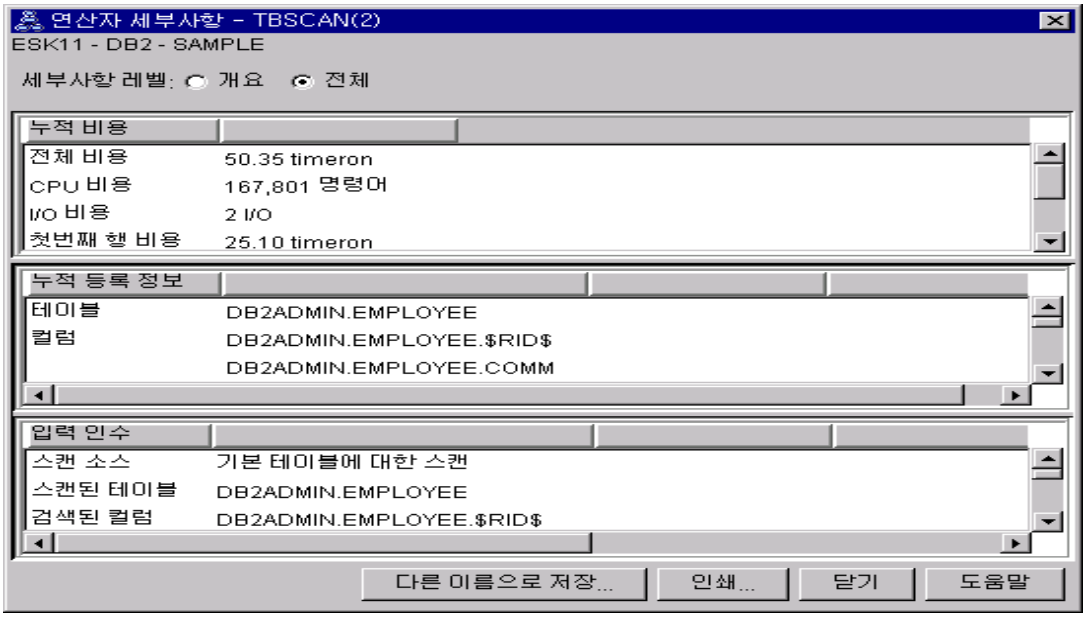


그림 4. Visual Explain : 연산자(Operator) 세부사항

이 창은 몇 개의 다른 섹션을 포함한다.

- 누적 비용 – 시스템 카탈로그 테이블에 저장된 통계치를 사용하여 계산된 예측된 누적비용
- 누적 등록 정보 – 쿼리를 만족하는 테이블, 컬럼 등에 대한 정보
- 입력 인수 – 연산자의 행위에 영향을 미치는 입력 인수에 대한 정보

Operand에 대한 자세한 정보를 조사하는 것도 가능하다. Operand 노드를 선택한 후, 노드 메뉴 항목에서 통계 표시를 선택한다. 그림 5는 DB2ADMIN.EMPLOYEE 테이블에 대한 자세한 Operand 내용을 보여준다.

그림 5. Visual Explain : Operand에 대한 자세한 통계 정보

Operand 노드에 대한 자세한 정보는 테이블 공간 정보, 오브젝트의 컬럼 갯수, 행의 갯수를 포함한 테이블 또는 색인 통계를 보여준다. 그림 5는 시스템 카탈로그 테이블에서 Explain과 현재의 통계치를 보여준다. 이들 통계치는 DB2 옵티마이저가 액세스 플랜을 결정하기 위해 사용한다. 그림 5에서는 DB2ADMIN.EMPLOYEE 테이블에 대해 수집된 통계값이 없음을 보여준다.

옵티마이저는 테이블에 대한 통계치를 가지고 있지 않을 때나 만약 테이블에 대한 통계치가 테이블의 행의 갯수(Cardinality)가 상대적으로 작다고 가리키면, 옵티마이저 자체가 테이블의 행의 갯수(Cardinality)를 계산하고자 한다. 옵티마이저는 테이블의 평균 컬럼 길이와 테이블이 사용하는 페이지수를 포함한 요소를 사용하여 이를 수행한다.
현재의 통계치는 좋은 액세스 플랜에서의 주요한 열쇠가 된다. 만약 DB2가 쿼리에 포함된 오브젝트들의 특성들을 알지 못한다면, 좋은 액세스 플랜을 생성하지 못할 수도 있다. 가장 최근의 통계치를 옵티마이저가 사용하려면, DB2 유틸리티를 사용해야만 한다. 이 유틸리티를 RUNSTATS라고 불리운다. 다음은 DB2ADMIN.EMPLOYEE 테이블에 대한 통계치를 수집하는 예이다.

- RUNSTATS ON TABLE DB2ADMIN.EMPLOYEE
  WITH DISTRIBUTION AND DETAILED INDEXES ALL

DB2ADMIN.EMPLOYEE 테이블에 대한 통계치는 시스템 카탈로그 테이블에 저장된다. RUNSTATS 유틸리티를 수행한 후, 데이터베이스에 대해 패키지를 리바인드(Rebind)하고 SQL 문장의 Explain을 다시 해야 한다. 현재의 통계치에 대한 값이 변경되고, 생성된 액세스 플랜의 전체 비용도 변경된다. 그림 6은 갱신한 DB2ADMIN.EMPLOYEE 통계치가 변경되었슴을 보여준다.

그림 6. Visual Explain : RUNSTATS 수행후의 테이블 통계

동적 SQL 문장에 대한 액세스 플랜을 결정할 때, DB2 옵티마이저는 항상 현재의 통계치를 사용한다. 정적 SQL 문장에 대해서는, DB2는 BIND 시에 통계치를 사용한다. 통계치가 갱신되기 전에 컴파일되었던 정적 SQL 문장들에 대해 현재의 통계치를 사용하기 위해서는 패키지는 재생성되어야 한다. 이것은 REBIND 명령어를 사용하여 수행될 수 있다.

4.10 Index Adivisor

Index Advisor는 테이블에 색인을 디자인하는 데 도움을 제공하는 관리 툴이다. 이것은 다음의 상황에 유용하다.

- 문제의 쿼리에 대해 최적의 색인들을 찾고자 할 때
- 선택적으로 적용하는 자원 한계에 종속된 쿼리의 집합(워크 로드)에 대한 최적의 색인들을 찾고자 할 때
- 색인을 생성하지 않고 워크 로드에서 색인을 테스트하고자 할 때

이 툴과 관련되어 두가지 개념이 있다. : 워크 로드와 가상 색인

워크 로드는 DB2가 주어진 시간에 처리해야 하는 SQL 문장들 (SELECT, INSERT, UPDATE, DELETE)의 집합이다. 워크 로드의 정보는 주어진 시간에서의 SQL 문장들의 유형과 빈도수와 관련되어 있다. Index Advisor는 색인을 추천하기 위해서 데이터베이스 정보와 함께 워크 로드 정보를 사용한다. 가상 색인은 현재의 데이터베이스 스키마에 존재하지 않는 색인이다.

Index Advisor 툴은 EXPLAIN 테이블의 확장인 두개의 테이블들을 사용한다.

- ADVISE_WORKLOAD

  이 테이블은 워크 로드를 설명한다. 테이블내의 각 행은 SQL 문장을 나타내고, 관련된 빈도수에 의해 설명된다. WORKLOAD_NAME이라 불리우는 테이블의 필드에는 각 워크 로드에 대한 식별자가 존재한다. 같은 워크 로드에 속하는 모든 SQL 문장들은 같은 WORKLOAD_NAME을 가져야 한다.

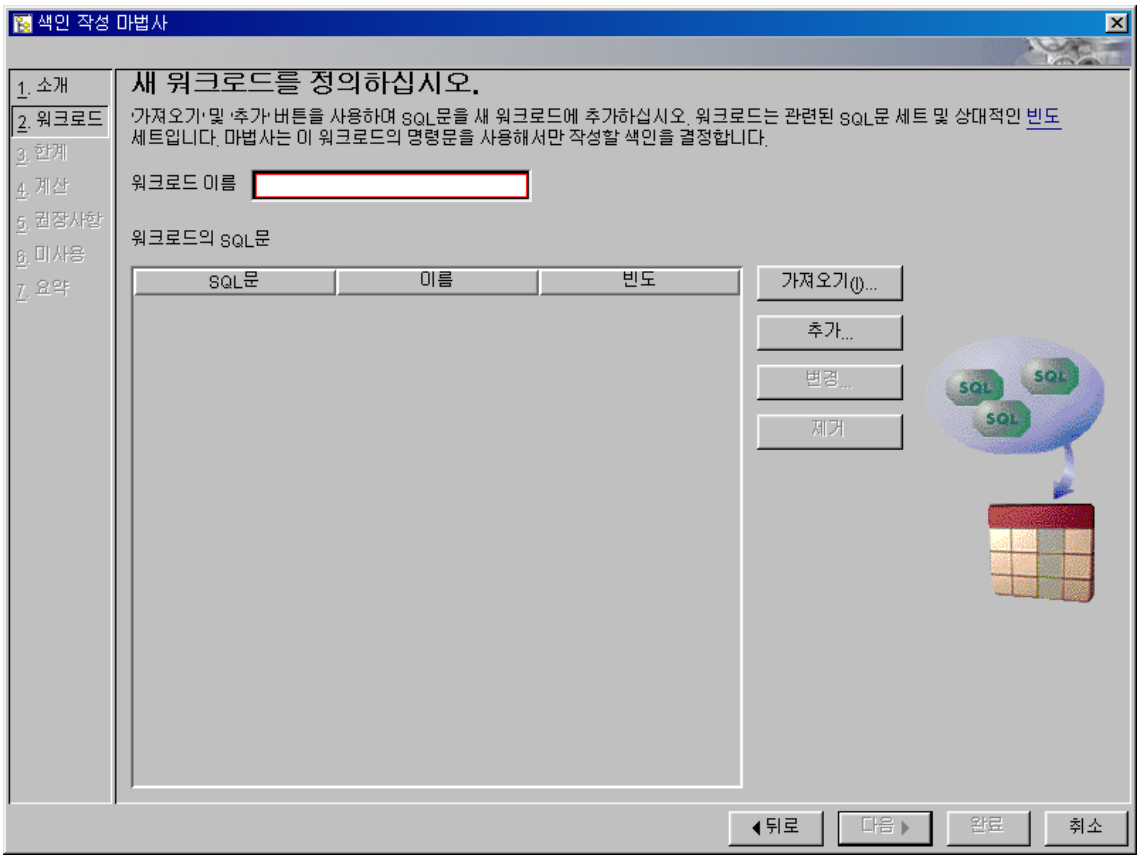
- ADVISE_INDEX

  이 테이블은 권장하는 색인들에 대한 정보를 저장한다. SQL 컴파일러, 색인 작성 마법사, db2advis 툴(Index Advisor), SQL 문장을 사용하여 매뉴얼 방식으로 정보가 이 테이블에 놓여진다.

CURRENT EXPLAIN MODE 특수 레지스터에 RECOMMEND INDEXES를 설정함으로써, EXPLAIN 함수를 호출할 때, ADVISE_INDEX 테이블이 생성된다.

CURRENT EXPLAIN MODE 특수 레지스터에 EVALUATE INDEXES를 설정하면, EXPLAIN 과정에서는 ADVISE_INDEX 테이블이 입력값으로 사용되어, 가상 색인 정의를 읽고 마치 실제 색인이 있는 것처럼 사용된다.

Index Advisor는 db2advis 유틸리티나 제어 센터에서 마법사를 사용한 색인을 선택함으로써 호출될 수 있으며, 그 화면은 다음과 같습니다. 이 화면에 사용할 SQL문과 그 빈도수를 입력하면, DB2 옵티마이저가 최적의 인덱스를 선택하여 보여줄 것이다.



4.11 DB2 조정자(governor)

DB2를 모니터링할 때, 병목현상이 시스템의 어디에서 발생하는지, 데이터베이스 활동의 어떤 유형이 발생되는지 등을 감지할 수 있다. 만약 어떤 응용프로그램이 많은 자원들을 필요로 하는지 등을 알기 위해서는 데이터베이스 응용프로그램의 행동방식을 분석해야 한다. 따라서DBA는 먼저 모니터링 기술을 사용하여 응용프로그램을 감지하고 나서 응용프로그램의 행동방식을 변경하거나, 시스템에서 응용프로그램을 중지해야 한다.

DB2 조정자는 자동적으로 그러한 확인을 수행하는 서버 응용프로그램이다. 또한 조정자는 서버에서 매우 많은 자원들을 사용한다고 생각되는 응용프로그램을 중지할 수 있다. (인스턴스에서 하나 또는 모든 응용프로그램을 중지하기 위해 DB2의 FORCE APPLICATION 명령어를 사용한다.)

조정자는 데이터베이스에 대해 수행하는 응용프로그램들에 대한 통계치를 수집한다. 그리고 나서 데이터베이스에 대해 지정한 규칙에 대해 이 통계치들을 확인한다. 지정한 규칙들의 예는 다음과 같다.

- 항상 빠른 시간에 완료되도록 응용프로그램 X의 우선순위를 증가한다.
- 응용프로그램들의 부분집합 즉 A, B, C의 속도를 늦춘다.
- 15분 이상의 작업단위(UOW)를 수행하지 않도록 한다.

조정자는 지정한 응용프로그램들에 대해 매개변수들을 변경하거나 시스템에서 응용프로그램을 중지함으로써 이들 규칙들을 적용시킨다. DB2 모니터링의 시작과 중지하는 같은 방법으로 DB2 조정자를 시작, 중지한다. 예를 들어, mygov.cfg라 불리우는 조정자 구성 파일(조정자 규칙들을 포함하는)을 가지고 운영체제 명령어 창에서 db2gov 유틸리티를 사용하여, SAMPLE 데이터베이스를 모니터링하는 조정자를 시작할 수 있다.

- db2gov START SAMPLE mygov.cfg mygov.log

지정한 파일 이름, mygov.log는 DB2 조정자가 수행하는 활동들을 로깅하는 파일이다. 또한 SAMPLE 데이터베이스에 대해 수행하는 조정자는 다음과 같이 중지할 수 있다.

- db2gov STOP SAMPLE

복수의 데이터베이스에 대해, 수행하는 DB2 조정자의 복수 인스턴스들을 가질 수 있다. DB2 조정자는 일정 간격으로 통계치를 수집하고 데이터베이스 응용프로그램의 활동을 모니터하기 때문에 수행시DB2 데이터베이스에 대해 성능에 영향을 줄 수 있다. db2govlg 툴을 사용하여 DB2 조정자에 의해 생성되는 로그들을 조사할 수 있다.