UNIT 14

모니터링

데이터베이스와 응용프로그램에 대한 모니터링은 데이터베이스의 효율적인 운영과 장애 예방을 대비하는 중요한 대책입니다. DB2는 특정 시점의 모니터링 정보를 위한 스냅샷 모니터와 스냅샷 함수를 제공하며, 일정 기간 동안의 모니터링 정보 수집을 위한 이벤트 모니터를 제공합니다.

DB2 9.7 운영자 가이드

Administrator Edition

- 오류 진단 파일
- 시스템 모니터 스위치
- 세션별 모니터 스위치
- 스냅샷 모니터
- 스냅샷 테이블 함수
- 응용프로그램 목록
- 응용프로그램이 사용한 CPU 시간
- 응용프로그램이 처리한 행의 수
- 응용프로그램별 잠금
- 파티션별 잠금
- 테이블별 잠금
- 잠금 대기 에이전트
- 잠금 대기 에이전트의 정적 SQL문
- 잠금 대기 에이전트의 동적 SQL문
- 잠금 보유 에이전트의 정적 SQL문
- 잠금 보유 에이전트의 동적 SQL문
- 응용프로그램별 로그 사용량
- 데이터베이스별 로그 사용량
- 테이블 스페이스 사용량
- 테이블 스페이스 컨테이너 사용량
- 테이블 스페이스 적중률
- 이벤트 모니터
- CREATE EVENT MONITOR 문
- 파일 이벤트 모니터
- 테이블 이벤트 모니터
- 시간소요 모니터
- db2pd 모니터링
- db2top
- db2top - Application
- db2top - Memory
- db2top - Lock
- db2top - Table
- db2top - Partitioning
- db2top - Dynamic SQL
- db2top - Utility
- db2top - Tablespace

TOPIC 14-01 오류 진단 파일

Point

데이터베이스 시스템 오류가 발생하면 db2diag.log 라는 파일에 기록됩니다. DIAGLEVEL과 DIAGPATH 인스턴스 구성 변수를 이용하여 기록할 오류의 수준과 진단 파일을 저장할 디렉토리를 결정합니다.

Tip

db2diag.log가 생성되는 기본 디렉토리는 <인스턴스의 홈디렉토리>/sqllib/db2dump입니다.

Tip

DIAGLEVEL의 기본값은 3으로 심각한 오류, 일반 오류, 경고 메시지를 기록합니다.

Tip

db2diag.log 를 비롯한 진단 파일은 삭제해도 무방합니다. 필요시에 자동적으로 새로운db2diag.log 파일이 다시 생성됩니다.

1 인스턴스 사용자로 로그인하여 db2diag.log 파일을 확인합니다.

```
$ login <인스턴스 사용자명>
$ more  $HOME/sqllib/db2dump/db2diag.log
```

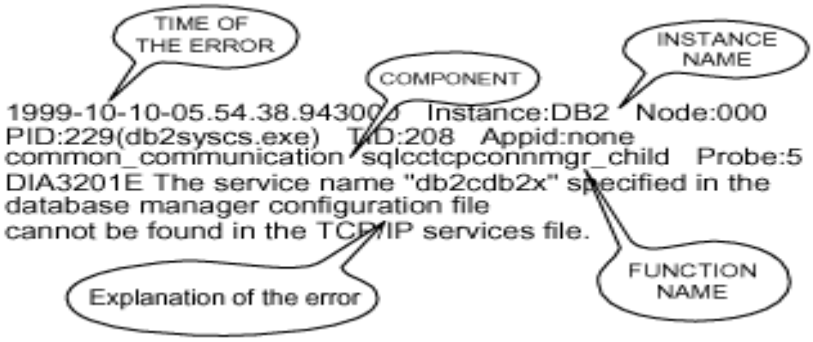


Figure 1401A ••• db2diag.log 파일

Tip

DIAGPATH 구성 변수에서 지정한 디렉토리는 미리 생성되어야 합니다.

2 update dbm cfg 명령어로 인스턴스 구성 변수인 DIAGLEVEL과 DIAGPATH를 변경하면, db2diag.log 에 기록되는 진단 정보의 수준과 db2diag.log 를 비롯한 시스템 오류 덤프 파일이 저장되는 디렉토리가 변경됩니다.

```
$ db2 attach to <인스턴스명>
$ db2 update dbm cfg using DIAGLEVEL <진단 수준>
$ db2 update dbm cfg using DIAGPATH <진단 디렉토리명>
$ db2 get dbm cfg | grep DIAG
```

Tip

심각한 오류가 발생한 경우에는 trap(tnnnnn.000), dump(nnnnn.dmp), core 파일 등이 추가적으로 생성됩니다.

Tip

IBM에 문제 해결을 요청할 때는 반드시 db2diag.log를 비롯한 trap, dump, core 파일 등을 보관하고 있어야 합니다.

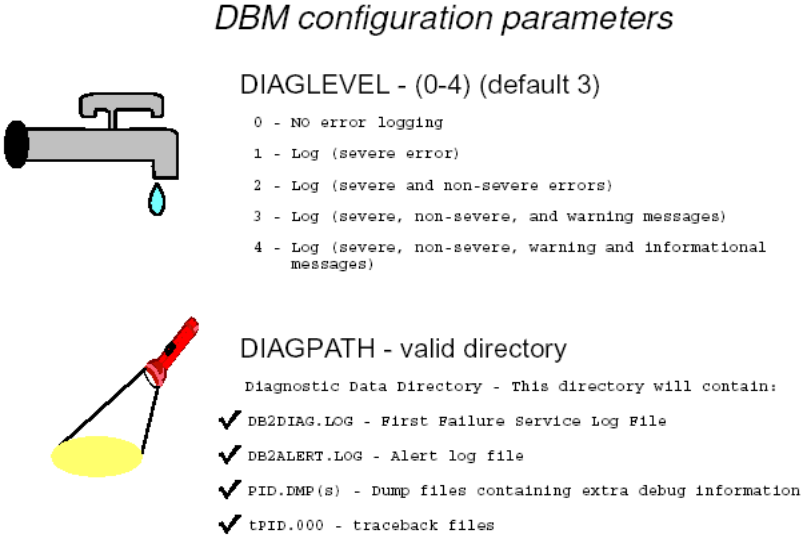


Figure 1401B ••• DIAGLEVEL과 DIAGPATH 인스턴스 구성 변수

TOPIC 1401 오류 진단 파일

Point

파일시스템 저장공간이 가득 차는 것을 방지하기 위하여 db2diag.log 파일의 크기를 제한할 수 있습니다.

1 아래의 명령을 수행하면 1024 메가 바이트 중 diaglog파일이 90%, db2<instance>.nfy파일이 10%의 비율로 생성되며 파일사이즈의 합이 1024 메가 바이트로 파일들이 생성됩니다. 파일은 rotating 방식으로 생성되며 db2diag.0.log 부터 순차적으로 10개의 파일이 생성됩니다.

```
$ db2 update dbm cfg using DIAGSIZE 1024
```

```
$ ls -l
total 8696
-rw-rw-rw- 1 db21r1 dblr1adm 172781     2009-06-03 10:02 db2diag.0.log
-rw-rw-rw- 1 db21r1 dblr1adm 172839     2009-06-03 10:07 db2diag.1.log
-rw-rw-rw- 1 db21r1 dblr1adm 172716     2009-06-03 10:12 db2diag.2.log
-rw-rw-rw- 1 db21r1 dblr1adm 172360     2009-06-03 10:18 db2diag.3.log
-rw-rw-rw- 1 db21r1 dblr1adm 172862     2009-06-03 10:23 db2diag.4.log
-rw-rw-rw- 1 db21r1 dblr1adm 172635     2009-06-03 12:07 db2diag.5.log
-rw-rw-rw- 1 db21r1 dblr1adm 93338      2009-06-03 14:16 db2diag.6.log
-rw-rw-rw- 1 db21r1 dblr1adm 3362       2009-06-03 09:21 db2diag.log
-rw-r----- 1 db21r1 dblr1adm 6291312    2009-06-03 14:16 db2eventlog.000
-rw-rw-rw- 1 db21r1 dblr1adm 2436       2009-06-03 14:16 db21r1.0.nfy
-rw-rw-rw- 1 db21r1 dblr1adm 1074       2009-06-03 09:21 db21r1.nfy
drwxrwxr-t 2 db21r1 dblr1adm 4096       2009-06-03 09:47 events
drwxrwxr-t 2 db21r1 dblr1adm 4096       2009-06-03 09:46 stmmlog
```

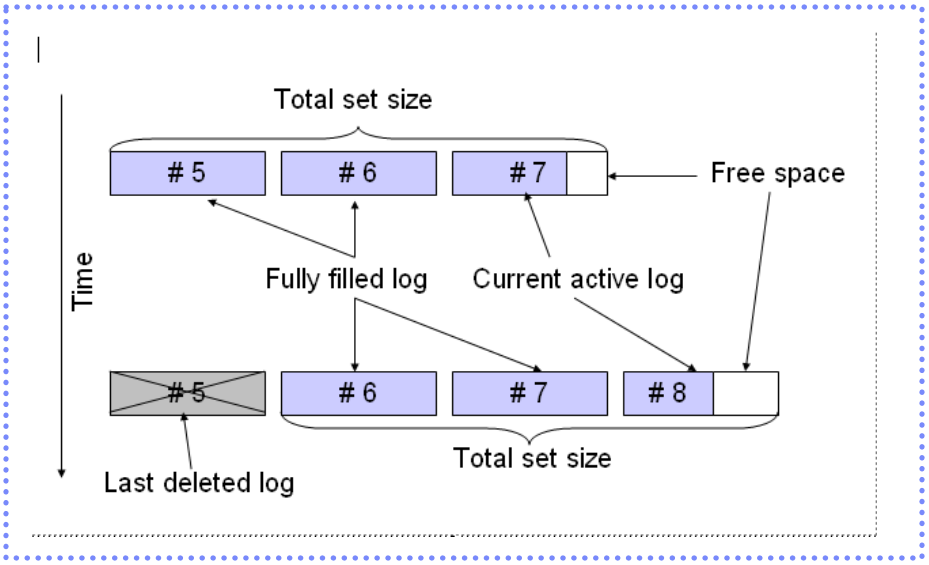


Figure 1401B • • • Rotating Diagnostic logs

TOPIC 14-02 시스템 모니터 스위치

Point

모니터링 자료의 수집 여부를 결정하는 시스템 모니터 스위치는 7가지 인스턴스 구성 변수를 이용하여 조절합니다. 인스턴스를 생성한 직후에는 DFT_MON_TIMESTAMP 구성 변수를 제외한 다른 모니터 스위치의 기본값은 OFF 입니다.

Tip

DFT_MON_TIMESTAMP는 DB2 엔진이 운영 체제를 호출하여 모든 명령문의 실행 전후에 시간소인을 확보하도록 하는 스위치로 많은 비용을 소모합니다. CPU 사용율이 심하게 높은 경우에는 OFF로 설정합니다.

Tip

모든 모니터링 응용프로그램은 응용프로그램이 첫 번째 모니터링 요청을 발행할 때 기본 스위치 설정값을 적용하고, 그 설정값을 계속 상속합니다.

1 get dbm cfg 명령어를 이용하여 기본 모니터 스위치의 현재 설정값을 확인합니다. DFT_MON_BUFPOOL, DFT_MON_LOCK, DFT_MON_SORT, DFT_MON_STMT, DFT_MON_TABLE, DFT_MON_UOW, DFT_MON_TIMESTAMP 등의 7가지 기본 모니터 스위치가 제공됩니다.

```
$ login <인스턴스 사용자>
$ db2 get dbm cfg | grep DFT
```

2 시스템 모니터 스위치를 활성화시키려면 update dbm cfg 명령어로 모니터 스위치의 값을 ON 또는 OFF 로 설정합니다. 변경 이후에 시작하는 모든 세션에는 변경된 시스템 모니터 스위치의 값이 적용됩니다. 값이 ON 인 경우에는 해당 항목에 대한 추가적인 모니터링 자료가 수집되고, OFF 인 경우에는 추가적인 모니터링 자료에 대한 수집이 중지됩니다.

```
$ db2 attach to <인스턴스명>
$ db2 update dbm cfg using <기본 모니터 스위치 변수명> ON
$ db2 update dbm cfg using <기본 모니터 스위치 변수명> OFF
$ db2 detach
```

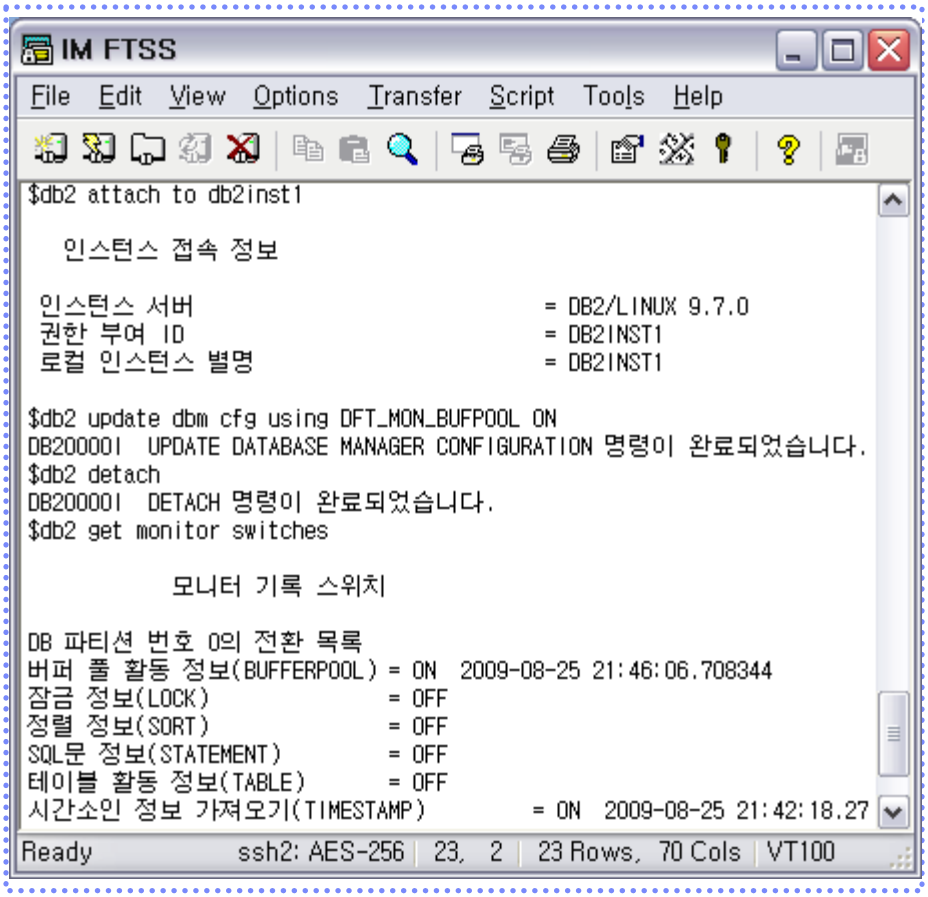


Figure 1402A ••• 시스템 모니터 스위치의 설정값

TOPIC 1403 세션별 모니터 스위치

Point

UPDATE MONITOR SWITCHES 명령어로 현재 세션의 모니터 스위치의 값을 설정하면, 인스턴스 구성 변수에 설정된 시스템 모니터 스위치의 값보다 우선적으로 적용됩니다.

Tip

- 변경된 모니터 스위치의 값은 현재의 세션에만 적용됩니다. terminate 명령으로 세션이 종료되면 모니터 스위치는 기본값으로 복원됩니다.

1. get monitor switches 명령어를 이용하여 모니터링 스위치의 현재 설정값을 확인합니다.

```
$ login <인스턴스 사용자>
$ db2 get monitor switches
```

2. 새로운 세션을 시작하기 전에 update monitor switches 명령어로 7가지의 모니터 스위치 중에서 원하는 모니터 스위치를 ON 또는 OFF 로 설정합니다.

```
$ db2 update monitor switches using <모니터 스위치명> ON
$ db2 update monitor switches using <모니터 스위치명> OFF
$ db2 get monitor switches
```

3. reset monitor 명령어는 모니터링 정보 중에서 엔진이 관리하는 내부적인 누적값을 표시하는 항목의 현재값을 초기값으로 갱신합니다.

```
$ db2 reset monitor all
$ db2 reset monitor for db <데이터베이스명>
```

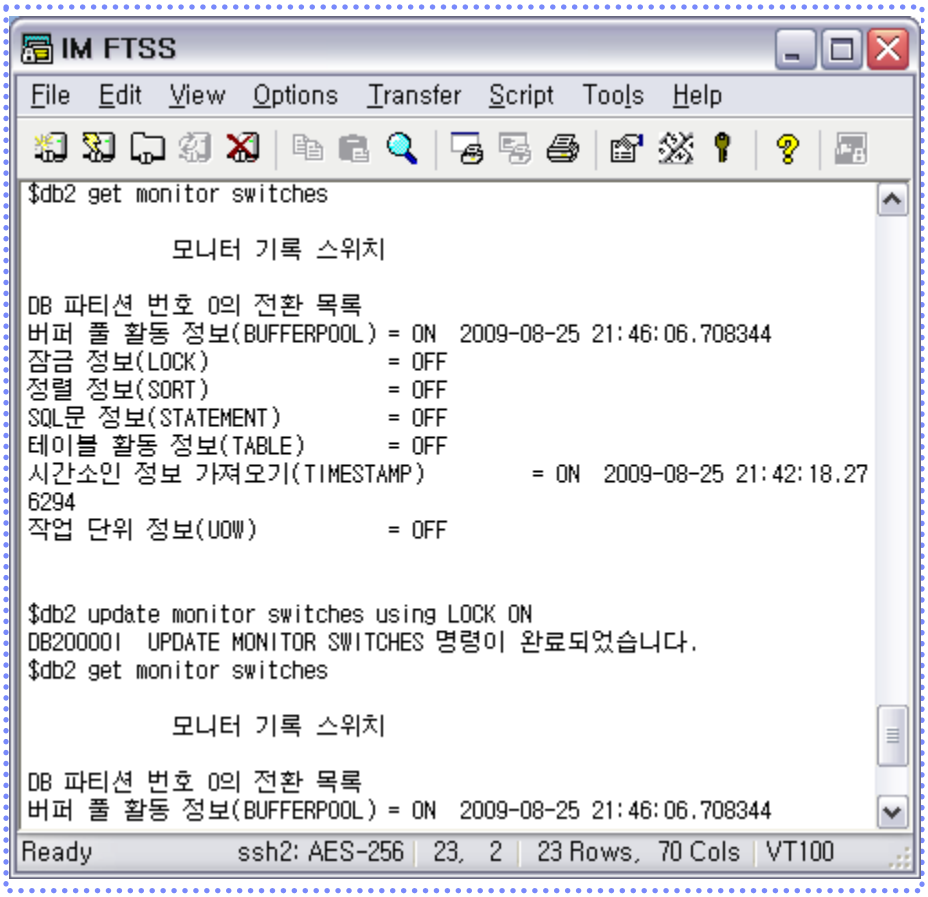


Figure 1403A ••• 현재 세션의 모니터 스위치 설정값

TOPIC 1404 스냅샷 모니터

Point

GET SNAPSHOT 명령어를 이용하여 현재 시점의 데이터베이스의 활동 상황을 모니터링할 수 있습니다. 모니터링의 대상은 인스턴스, 데이터베이스, 응용프로그램이 될 수 있습니다. 세션의 모니터 스위치의 설정값에 따라 표시되는 텍스트 정보가 달라집니다.

Tip

인스턴스, 데이터베이스, 응용프로그램, 테이블 스페이스, 테이블, 잠금, 버퍼풀, 동적 SQL문 등에 대한 모니터링 정보가 수집됩니다.

Tip

모니터링 항목은 현재값 또는 누적값을 표시합니다.

New

9.7에서는 Health Monitor는 더 이상 지원하지 않습니다. (첨부참고)

1. 인스턴스 사용자로 로그인합니다.

```
$ login <인스턴스 사용자명>
```

2. get snapshot 명령어를 이용하여 인스턴스의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 get snapshot for dbm
```

3. get snapshot 명령어를 이용하여 데이터베이스의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 get snapshot for all on <데이터베이스명>
$ db2 get snapshot for db on <데이터베이스명>
$ db2 get snapshot for locks on <데이터베이스명>
```

4. get snapshot 명령어를 이용하여 응용프로그램의 활동 내역을 모니터링할 수 있습니다.

```
$ db2 list applications
$ db2 get snapshot for application agentid  <응용프로그램 핸들 번호>
```

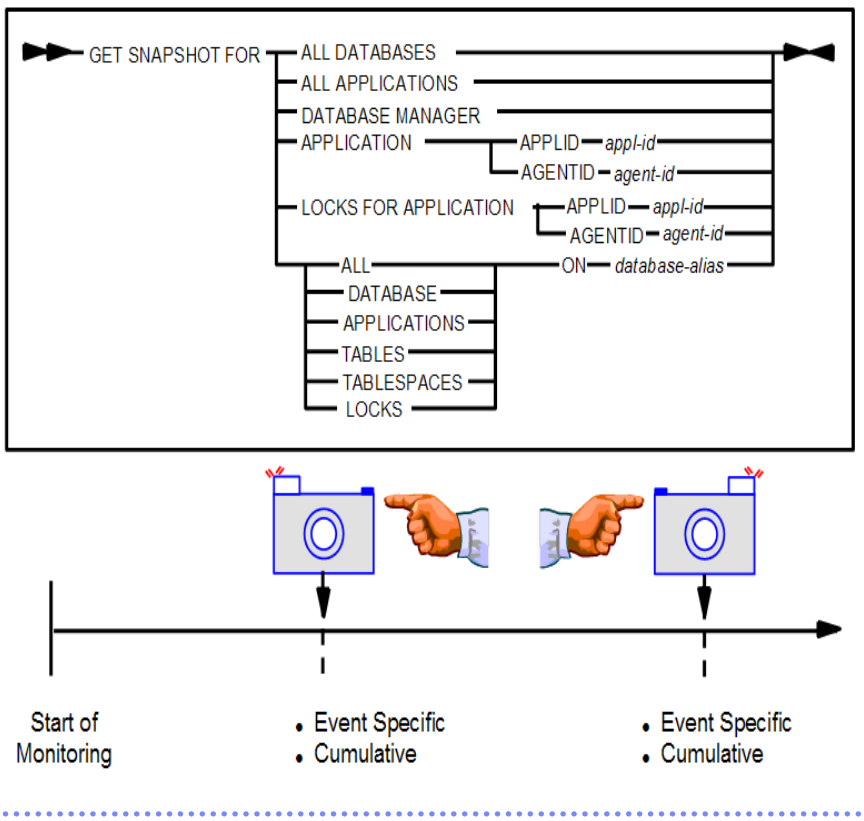


Figure 1404A... GET SNAPSHOT 명령어

TOPIC 1405 스냅샷 테이블 함수

Point

GET SNAPSHOT 명령어의 실행 결과와 동일한 스냅샷 정보를 테이블의 형태로 저장하는 테이블 함수입니다. SELECT 문의 FROM 절에서 표현되고, 일반 테이블과 동일한 방법으로 조회할 수 있습니다.

1 대표적인 스냅샷 테이블 함수의 종류는 다음과 같습니다.

함 수	설 명
SNAPSHOT_AGENT	응용프로그램 스냅샷에서 응용프로그램의 핸들 번호와 대응되는 db2agent 프로세스의 pid를 반환합니다.
SNAPSHOT_APPL	응용프로그램 스냅샷에서 응용프로그램에 대한 일반적인 정보를 반환합니다.
SNAPSHOT_APPL_INFO	응용프로그램 스냅샷에서 응용프로그램의 핸들 번호와 이름, 응용프로그램 프로세스의 pid 등을 반환합니다.
SNAPSHOT_BP	버퍼풀 스냅샷에서 버퍼풀의 활동 정보를 반환합니다.
SNAPSHOT_CONTAINER	테이블 스냅샷에서 컨테이너 구성 정보를 반환합니다.
SNAPSHOT_DATABASE	데이터베이스 스냅샷에서 정보를 반환합니다.
SNAPSHOT_DBM	인스턴스 스냅샷에서 정보를 반환합니다.
SNAPSHOT_DYN_SQL	동적 SQL 스냅샷에서 정보를 반환합니다
SNAPSHOT_FCM	FCM에 관한 정보를 반환합니다.
SNAPSHOT_LOCK	잠금 스냅샷에서 정보를 반환합니다.
SNAPSHOT_LOCKWAIT	응용프로그램 스냅샷에서 잠금 대기 정보를 반환합니다.
SNAPSHOT_STATEMENT	응용프로그램 스냅샷에서 명령문의 정보를 반환합니다.
SNAPSHOT_TABLE	테이블 스냅샷에서 활동 정보를 반환합니다.
SNAPSHOT_TBS	테이블 스페이스 스냅샷에서 활동 정보를 반환합니다.
SNAPSHOT_TBS_CFG	테이블 스페이스 스냅샷에서 구성 정보를 반환합니다

2 get dbm cfg 명령어로 시스템 모니터 스위치의 기본값을 확인합니다.

```
$ db2 get dbm cfg | grep DFT_MON
```

3 get monitor switches 명령어를 이용하여 현재 세션의 모니터링 스위치의 값을 확인합니다.

```
$ db2 get monitor switches
```

4 SELECT문의 FROM 절에서 TABLE 이라는 키워드를 사용하여 스냅샷 함수를 실행합니다. 인수로 <데이터베이스명>과 <데이터베이스 파티션 번호>를 입력합니다. 파티션 번호가 -1 이면 현재 파티션을 의미하고, -2 이면 모든 파티션을 의미합니다. <결과 테이블에 대한 별명>은 임의로 지정합니다.

```
SELECT * FROM TABLE(SNAPSHOT_APPL_INFO('SAMPLE',-1)) TFSNAP
```

TABLE() function — table function — alias

Figure 1405A ··· 스냅샷 테이블 함수의 사용 방법

Tip

스냅샷 함수의 실행 결과를 저장할 임시 테이블의 별명을 지정하지 않으면 SELECT문은 실패합니다.

Tip

결과로 반환된 테이블은 SELECT문이 실행되는 동안에만 존재합니다.

TOPIC 14-06 응용프로그램 목록 확인

Point

데이터베이스에 접속하고 있는 응용프로그램에 대한 정보를 확인합니다. 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 응용프로그램 ID, 상태, 에이전트 개수, 데이터베이스명, 프로세스 ID 를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_name,1,20) as appl_name
    ,substr(appl_info.appl_id,1,30) applid
    ,case appl_info.appl_status
        when 2 then 'Connection Completed'
        when 3 then 'Executing'
        when 4 then 'UOW Wait'
        when 5 then 'Lock Wait'
        when 9 then 'Compile'
        when 24 then 'Pending remote request'
        when 26 then 'Decoupled'
        else substr(char(appl_info.appl_status),1,10)
    end as status
    ,cast(appl.num_agents as integer) num_agents
    ,substr(appl_info.db_name,1,8) dbname
    ,cast(appl_info.client_pid as integer) client_pid
from    table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by appl.num_agents desc, appl_name, agentid;
```

2 실행 결과는 다음과 같습니다.

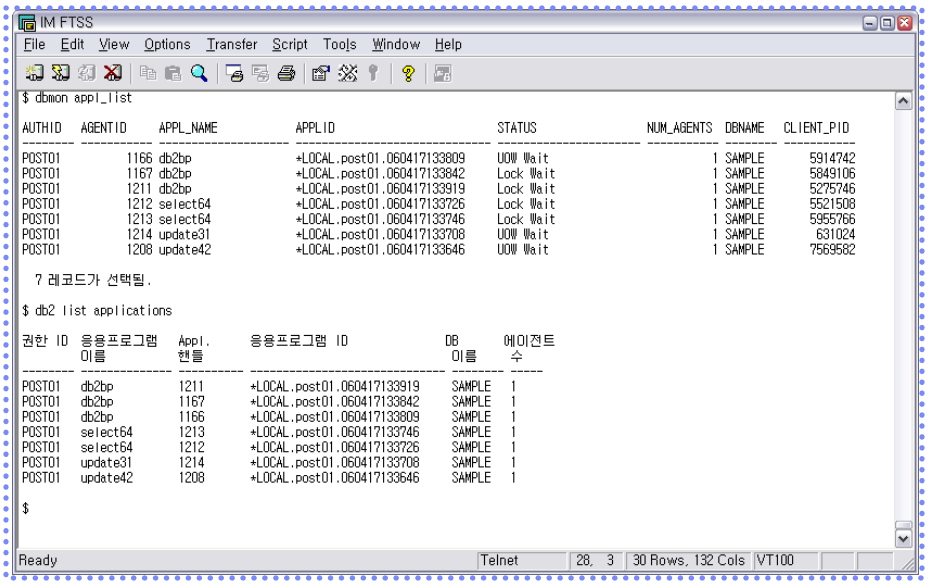


Figure 1406A ··· 응용프로그램 목록 확인

TOPIC 1407 응용프로그램이 사용한 CPU 시간

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, User CPU, System CPU, 경과 시간, 유휴 시간, 읽은 행수, 기록한 행수를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_info.appl_name,1,20) appl_name
    ,cast(appl.agent_usr_cpu_time_s as integer) as user_cpu
    ,cast(appl.agent_sys_cpu_time_s as integer) as sys_cpu
    ,timestampdiff( 2, char( current timestamp - appl.uow_start_time ))
    as elapsed_time
    ,cast(appl_idle_time as integer) idle_time
    ,appl.rows_read
    ,appl.rows_written
from table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
and uow_stop_time is null
order by user_cpu desc, agentid with ur;
```

2 실행 결과는 다음과 같습니다.

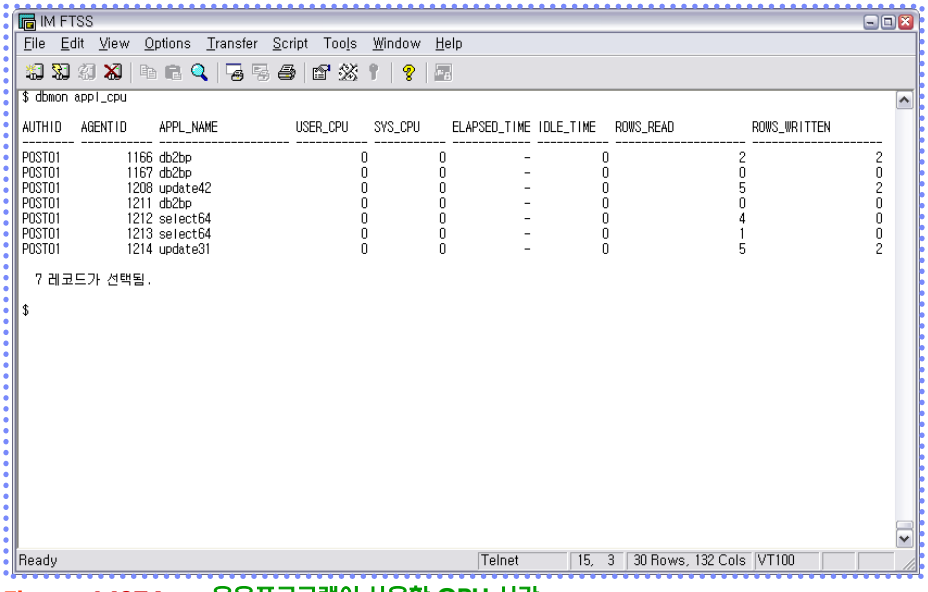


Figure 1407A ••• 응용프로그램이 사용한 CPU 시간

TOPIC 1408 응용프로그램이 처리한 행의 수

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 경과 시간, 조회 건수, 추가 건수, 갱신 건수, 삭제 건수, 읽은 행수, 기록한 행수를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_name,1,20) as appl_name
    ,timestampdiff( 2, char(current timestamp - uow_start_time))
    as elapsed_time
    ,cast(appl.rows_selected as integer) as rows_select
    ,cast(appl.rows_inserted as integer) as rows_insert
    ,cast(appl.rows_updated as integer) as rows_update
    ,cast(appl.rows_deleted as integer) as rows_delete
    ,cast(appl.rows_read as integer) rows_read
    ,cast(appl.rows_written as integer) rows_written
from table(snapshot_appl('$DBNAME', $DPMODE)) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
and uow_stop_time is null
order by elapsed_time desc, agentid with ur;
```

2 실행 결과는 다음과 같습니다.

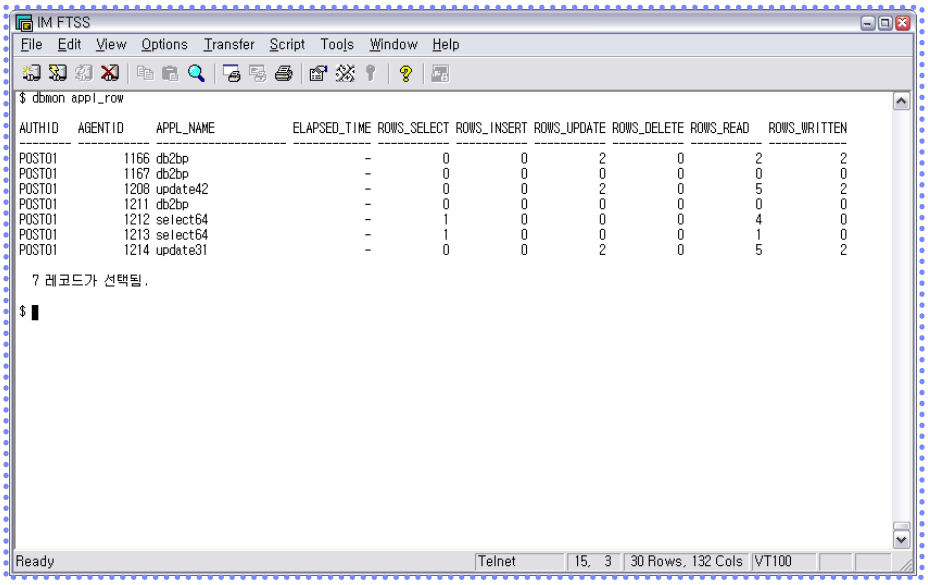


Figure 1408A ••• 응용프로그램이 처리한 행의 수

TOPIC 14-09 응용프로그램별 잠금

Point

응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 상태, 대기 잠금수, 보유 잠금수, escalation, 교착 상태 회수, 잠금 대기 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,14) as appl_name,
    case appl_info.appl_status
        when 2 then 'Connection Completed'
        when 3 then 'Executing'
        when 4 then 'UOW Wait'
        when 5 then 'Lock Wait'
        when 9 then 'Compile'
        when 24 then 'Pending remote request'
        when 26 then 'Decoupled'
        else substr(char(appl_info.appl_status),1,10)
    end as status,
    cast(appl.lock_waits as integer) as lock_waits,
    cast(appl.locks_held as integer) as locks_held,
    cast(appl.lock_escals as integer) as escals,
    cast(appl.x_lock_escals as integer) as x_escals,
    cast(appl.deadlocks as integer) as deadlock,
    cast(appl.locks_waiting as integer) as locks_waiting
from table(snapshot_appl('$DBNAME',$DPMODE)) as appl,
     table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid;
```

2 실행 결과는 다음과 같습니다.

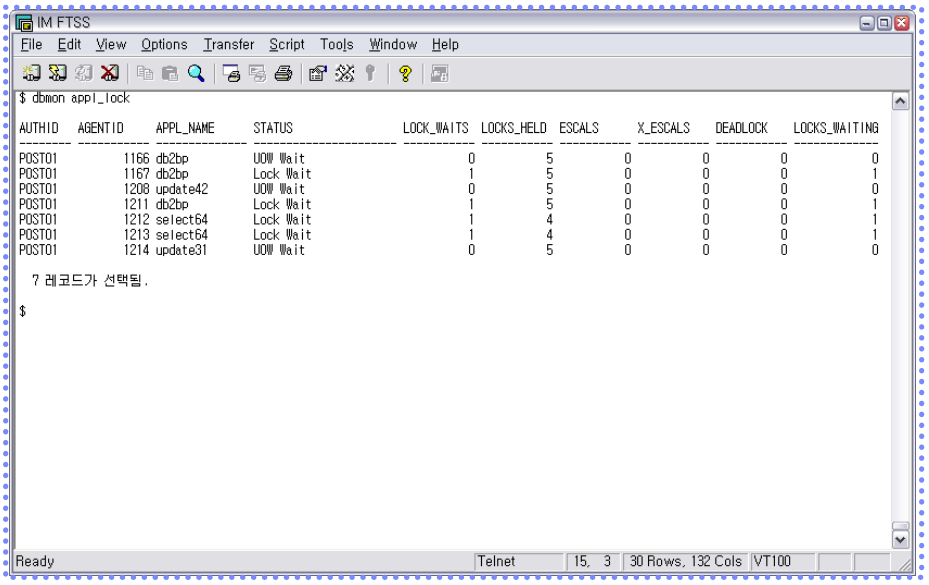


Figure 1409A ••• 응용프로그램별 잠금

TOPIC 14-10 파티션별 잠금

Point

특정 데이터베이스 파티션별로 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 테이블스페이스명, 테이블명, 잠금 수준, 잠금 대상, 잠금 모드, 잠금 상승 현상 발생 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 전체 파티션을 의미하는 -2 를 사용하지 말고, 특정 파티션 번호를 지정하도록 합니다.

Tip

단일 파티션을 사용한다면 <파티션번호> 에는 0 을 지정합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.AUTH_ID,1,10) as auth_id,
    cast(lock.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,16) as appl_name,
    substr(lock.tablespace_name,1,15) as tbsname,
    substr(lock.table_name,1,18) as tabname,
    lock.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row'
        else substr(char(lock.lock_object_type),1,4)
    end  as type,
    cast(lock.lock_object_name as integer) as lockobjname,
    case cast(lock.lock_mode as smallint)
        when 3 then 'S'
        when 5 then 'X'
        when 9 then 'NS'
        else substr(char(lock.lock_mode),1,4)
    end as mode,
    case lock.lock_status
        when 1 then 'G'
        when 2 then 'C'
        else cast(lock.lock_status as char)
    end as status,
    cast(lock.lock_escalation as integer) as escal
from    table(snapshot_lock('$DBNAME', $DPMODE )) as lock,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where lock.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid, tbsname, tabname;
```

2 실행 결과는 다음과 같습니다.

IM FTSS

File Edit View Options Transfer Script Tools Window Help

```
$ dbmon dp_lock 0

AUTH_ID    AGENTID    APPL_NAME        TBSNAME         TABNAME          TYPE                LOCKOBJNAME MODE STATUS ESCAL
---------- ---------- ---------------- --------------- ---------------- ------------------- ----------- ---- ------ ----------
POST01              7 update42         TS5             T1               Row                           5 X    G               0
POST01              7 update42         TS5             T1               Row                           7 X    G               0
POST01              7 update42         TS5             T1               Table                         2 IX   G               0
POST01              7 update42         -               -                Internal P Lock               0 S    G               0
POST01              7 update42         -               -                Internal P Lock               0 S    G               0
POST01              8 update31         TS5             T1               Row                           4 X    G               0
POST01              8 update31         TS5             T1               Row                           6 X    G               0
POST01              8 update31         TS5             T1               Table                         2 IX   G               0
POST01              8 update31         -               -                Internal P Lock               0 S    G               0
POST01              8 update31         -               -                Internal P Lock               0 S    G               0
POST01              9 select64         TS5             T1               Table                         2 IS   G               0
POST01              9 select64         -               -                Internal P Lock               0 S    G               0
POST01              9 select64         -               -                Internal P Lock               0 S    G               0
POST01             10 select53         TS5             T1               Table                         2 IS   G               0
POST01             10 select53         -               -                Internal P Lock               0 S    G               0
POST01             10 select53         -               -                Internal P Lock               0 S    G               0
POST01             11 db2bp            TS5             T1               Row                          10 X    G               0
POST01             11 db2bp            TS5             T1               Row                          11 X    G               0
POST01             11 db2bp            TS5             T1               Table                         2 IX   G               0
POST01             11 db2bp            -               -                Internal P Lock               0 S    G               0
POST01             11 db2bp            -               -                Internal P Lock               0 S    G               0
POST01             12 db2bp            TS5             T1               Table                         2 IS   G               0
POST01             12 db2bp            -               -                Internal V Lock               0 S    G               0
POST01             12 db2bp            -               -                Internal P Lock               0 S    G               0
POST01             12 db2bp            -               -                Internal P Lock               0 S    G               0
POST01             13 db2bp            TS5             T1               Table                         2 IX   G               0
```

Ready | Telnet | 30, 3 | 30 Rows, 132 Cols | VT100

Figure 1410A ··· 파티션별 잠금

TOPIC 14-11 테이블별 잠금

Point

특정 테이블별로 응용프로그램을 실행한 사용자, 핸들 번호, 응용프로그램명, 테이블스페이스명, 테이블명, 잠금 수준, 잠금 대상, 잠금 모드, 잠금 상승 현상 발생 여부 등을 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$TABNAME 은 <테이블명>으로 <스키마명>을 제외하고 지정합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.AUTH_ID,1,10) as auth_id,
    cast(lock.agent_id as integer) as agentid,
    substr(appl_info.appl_name,1,16) as appl_name,
    substr(lock.tablespace_name,1,15) as tbsname,
    substr(lock.table_name,1,18) as tabname,
    case lock.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row' as type,
    cast(lock.lock_object_name as integer) as lockobjname,
    case cast(lock.lock_mode as smallint)
        when 3 then 'S'
        when 5 then 'X'
        when 8 then 'U'
        when 9 then 'NS'
        else substr(char(lock.lock_mode),1,4)
    end as mode,
    case lock.lock_status
        when 1 then 'G'
        when 2 then 'C'
        else cast(lock.lock_status as char)
    end as status,
    cast(lock.lock_escalation as integer) as escal
from    table(snapshot_lock('$DBNAME', $DPMODE )) as lock,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where lock.agent_id = appl_info.agent_id
    and lock.table_name = ucase('$TABNAME')
    and appl_info.appl_id <> '$MON_APPL_ID'
order by agentid, tbsname, tabname, type;
```

2 실행 결과는 다음과 같습니다.

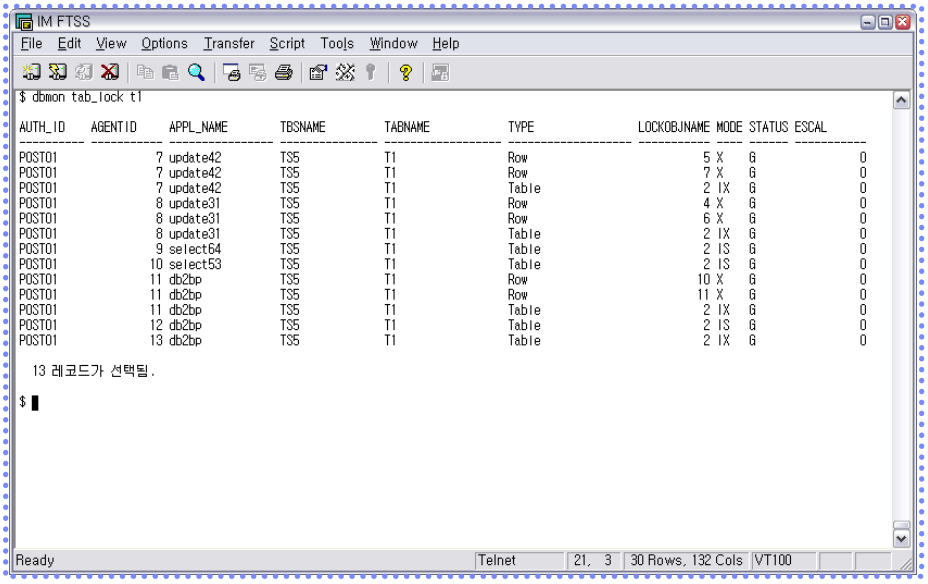


Figure 1411A ••• 테이블별 잠금

TOPIC 14-12 잠금 대기 에이전트

Point

잠금 대기 상태에 있는 에이전트의 목록을 확인합니다. 잠금 대기 응용프로그램명과 핸들 번호, 잠금 대상, 잠금 모드, 잠금을 보유하고 있는 응용프로그램명과 핸들 번호를 확인할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

잠금에 대한 정확한 스냅샷 정보를 확인하려면, 시스템 기본 모니터 스위치인 DFT_MON_LOCK을 ON으로 설정하는 것이 좋습니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) as wait_agent,
    substr(appl_info1.appl_name,1,10) as wait_appl,
    substr(lockwait.table_schema,1,18) as tabschema ,
    substr(lockwait.table_name,1,18) as tabname ,
    case lockwait.lock_object_type
        when 1 then 'Table'
        when 2 then 'Row'
        else substr(char(lockwait.lock_object_type),1,4)
    end as lock_type
    ,case cast(lockwait.lock_mode_requested  as smallint)
        when 3 then 'S'
        when 5 then 'X'
        else substr(char(lockwait.LOCK_MODE_REQUESTED),1,4)
    end as lock_mode
    ,cast(partition_number as smallint) partition
    ,cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info2.appl_name,1,10) as hold_appl
    ,timestampdiff( 2, char( current timestamp -
        lockwait.LOCK_WAIT_START_TIME)) as wait_time_s
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait,
    table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info1,
    table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info2
where
    lockwait.agent_id = appl_info1.agent_id
    and   lockwait.agent_id_holding_lk = appl_info2.agent_id
order by wait_agent,lock_type, hold_agent;
```

2 실행 결과는 다음과 같습니다.

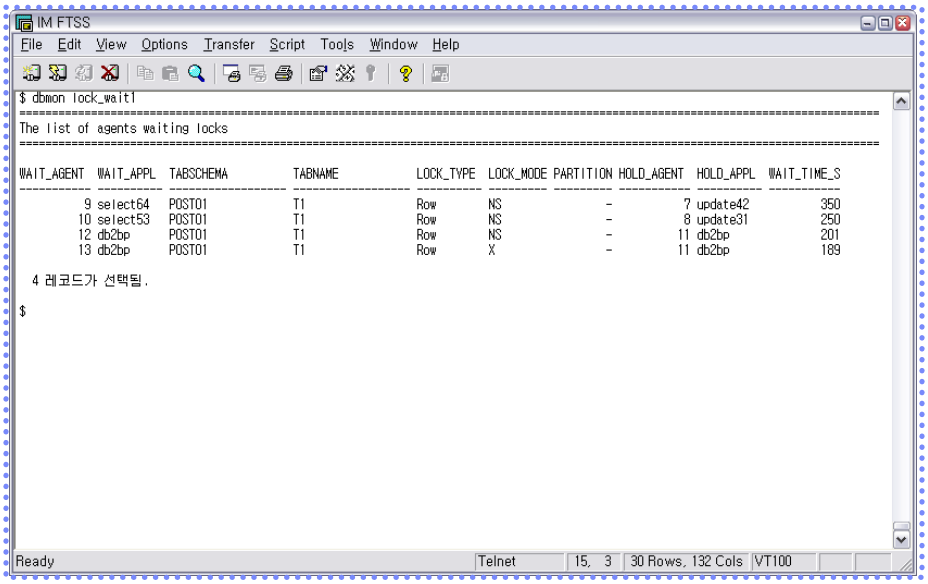


Figure 1412A ••• 잠금 대기 에이전트

TOPIC 14-13 잠금 대기 에이전트의 정적 SQL문

Point

잠금 대기 중인 에이전트가 실행할 정적 SQL문을 확인합니다. 해당 SQL문은 잠금을 보유한 응용프로그램이 UOW 를 종료할 때까지 실행되지 못하고 대기해야 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) wait_agent
    ,substr(appl_info.appl_name,1,10) wait_appl
    ,substr(statement.creator,1,8) pkg_schema
    ,substr(statement.package_name,1,8) package
    ,cast(statement.section_number as smallint) section
    ,substr(cat_statements.text,1,63) SQL
    ,cast(lockwait.agent_id_holding_lk as integer) hold_agent
from    table(snapshot_lockwait('$DBNAME',$DPMODE)) lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) statement
    ,syscat.statements cat_statements
where
    lockwait.agent_id = appl_info.agent_id                  and
    lockwait.agent_id = statement.agent_id                  and
    statement.stmt_text is null                             and
    cat_statements.pkgname = upper(statement.package_name)  and
    cat_statements.sectno = statement.section_number;
```

2 실행 결과는 다음과 같습니다.

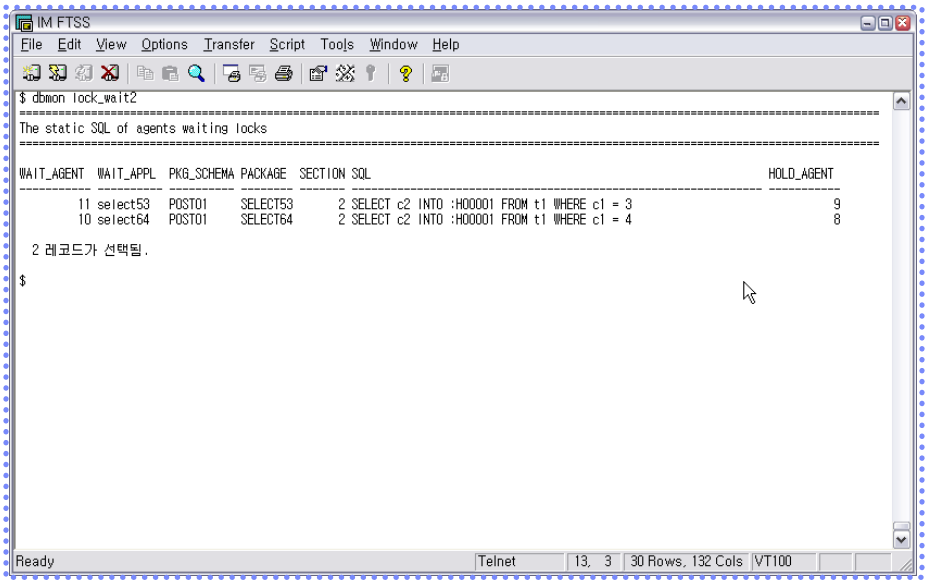


Figure 1413A ••• 잠금 대기 에이전트의 정적 SQL문

TOPIC 14-14 잠금 대기 에이전트의 동적 SQL문

Point

잠금 대기 중인 에이전트가 실행할 동적 SQL문을 확인합니다. 해당 SQL문은 잠금을 보유한 응용프로그램이 UOW 를 종료할 때까지 실행되지 못하고 대기해야 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2 를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id as integer) wait_agent
    ,substr(appl_info.appl_name,1,10) wait_appl
    ,cast(NULL as char(8)) pkg_schema
    ,cast(NULL as char(8)) package
    ,cast(NULL as smallint) section
    ,substr(statement.stmt_text,1,63) SQL
    ,cast(lockwait.agent_id_holding_lk as integer) hold_agent
from    table(snapshot_lockwait('$DBNAME',$DPMODE)) lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) statement
where
    lockwait.agent_id = appl_info.agent_id              and
    lockwait.agent_id = statement.agent_id              and
    statement.stmt_text is not null
order by wait_agent, hold_agent;
```

2 실행 결과는 다음과 같습니다.

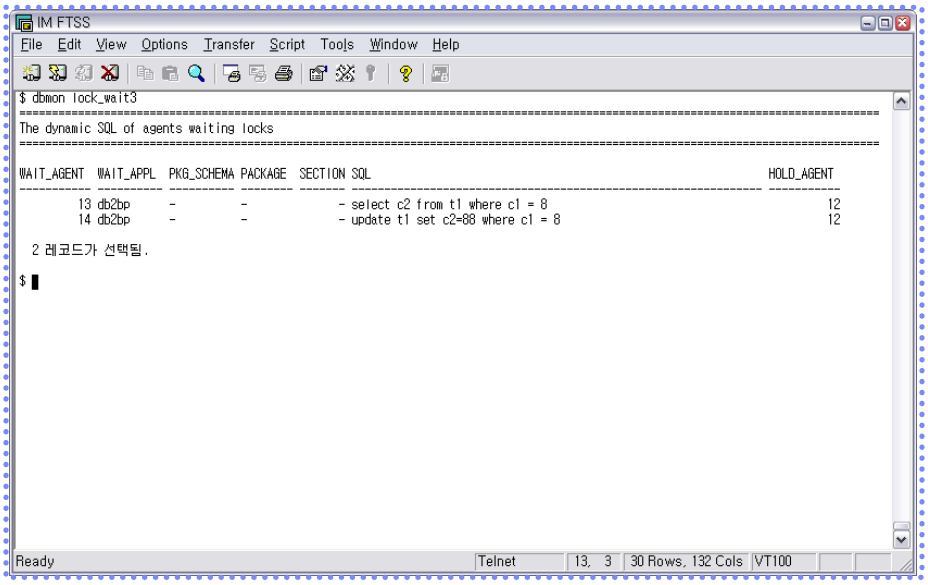


Figure 1414A ••• 잠금 대기 에이전트의 동적 SQL문

TOPIC 14-15 잠금 보유 에이전트의 정적 SQL문

Point

잠금을 보유하고 있는 에이전트가 실행한 정적 SQL 문을 확인합니다. 해당 SQL문의 실행이 완료되거나, UOW를 종료할 때까지 잠금을 보유합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select
    cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info.appl_name,1,10) as hold_appl
    ,substr(statement.creator,1,8) pkg_schema
    ,substr(statement.package_name,1,8) package
    ,cast(statement.section_number as smallint) last_section
    ,cast(cat_statements.sectno as smallint) section
    ,substr(cat_statements.text,1,61) last_SQL
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) as statement
    ,syscat.statements cat_statements
where
    lockwait.agent_id_holding_lk = appl_info.agent_id       and
    lockwait.agent_id_holding_lk = statement.agent_id       and
    cat_statements.pkgschema = upper(statement.creator)     and
    cat_statements.pkgname = upper(statement.package_name)  and
    cat_statements.sectno <= statement.section_number
order by hold_agent, section;
```

2 실행 결과는 다음과 같습니다.

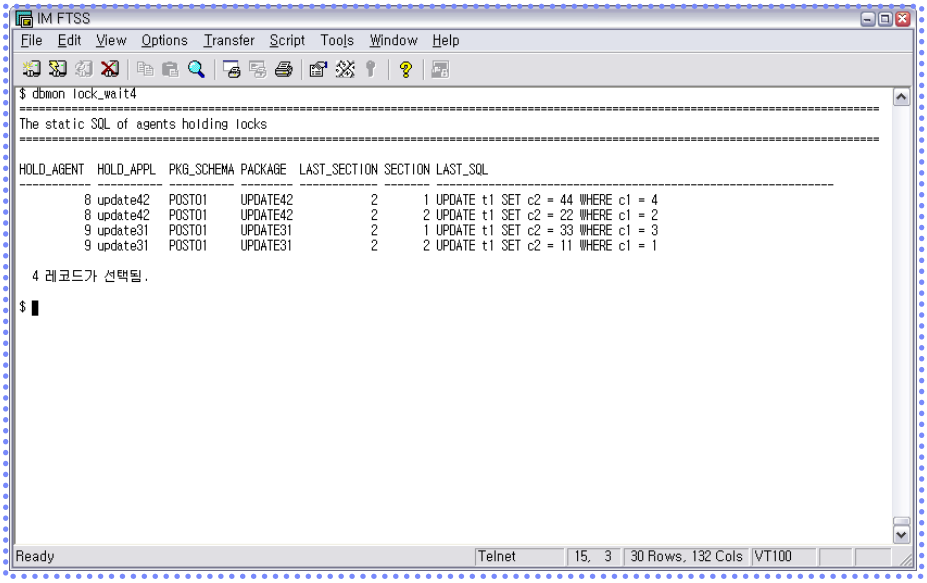


Figure 1415A • • • 잠금 보유 에이전트의 정적 SQL문

TOPIC 14-16 잠금 보유 에이전트의 동적 SQL문

Point

잠금을 보유하고 있는 에이전트가 실행한 마지막 SQL문을 확인합니다. 해당 SQL문의 실행이 완료되거나, UOW를 종료할 때까지 잠금을 보유합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select  distinct
    cast(lockwait.agent_id_holding_lk as integer) as hold_agent
    ,substr(appl_info.appl_name,1,10) as hold_appl
    ,cast(NULL as char(8)) pkg_schema
    ,cast(NULL as char(8)) package
    ,cast(NULL as smallint) last_section
    ,cast(NULL as smallint) section
    ,cast(substr(statement.stmt_text,1,61) as char(61)) SQL
from
    table(snapshot_lockwait('$DBNAME',$DPMODE)) as lockwait
    ,table(snapshot_appl_info('$DBNAME',$DPMODE)) as appl_info
    ,table(snapshot_statement('$DBNAME',$DPMODE)) as statement
where
    lockwait.agent_id_holding_lk = appl_info.agent_id       and
    lockwait.agent_id_holding_lk = statement.agent_id       and
    statement.stmt_text is not null
order by hold_agent, last_section;
```

2 실행 결과는 다음과 같습니다.

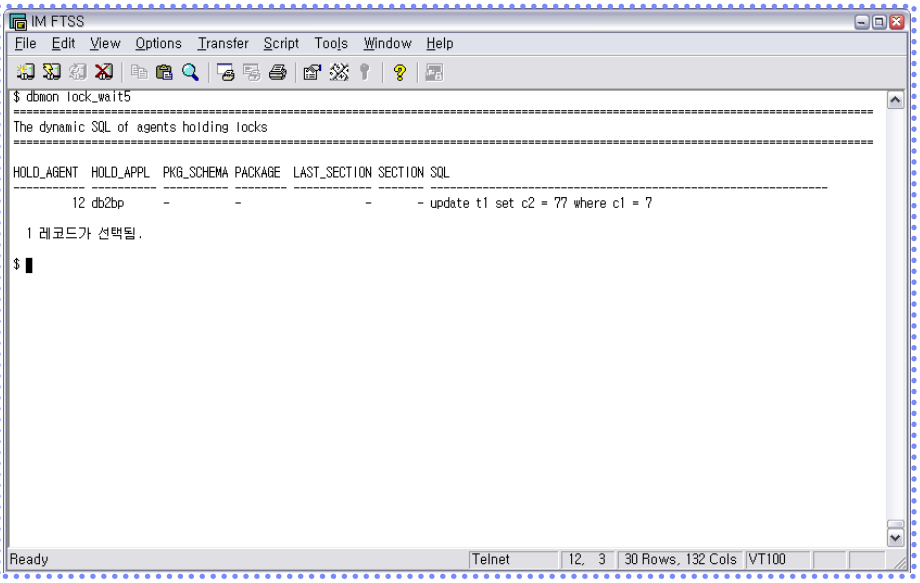


Figure 1416A ••• 잠금 보유 에이전트의 동적 SQL문

TOPIC 14-17 응용프로그램별 로그 사용량

Point

특정 응용프로그램이 사용하고 있는 로그의 사용량에 대한 정보를 확인합니다. 로그 사용량이 가장 많은 응용프로그램을 파악할 수 있습니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

Tip

$MON_APPL_ID는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(appl_info.auth_id,1,8) as authid
    ,cast(appl.agent_id as integer) as agentid
    ,substr(appl_info.appl_name,1,20) appl_name
    ,appl.uow_log_space_used
    ,cast(appl.agent_usr_cpu_time_s as integer) user_cpu
    ,timestampdiff( 2, char( current timestamp - appl.uow_start_time ))
        as elapsed_time
    ,appl.rows_read
    ,appl.rows_written
from table(snapshot_appl('$DBNAME', $DPMODE )) as appl,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where appl.agent_id = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID'
order by uow_log_space_used desc, rows_written desc, agentid;
```

2 실행 결과는 다음과 같습니다.

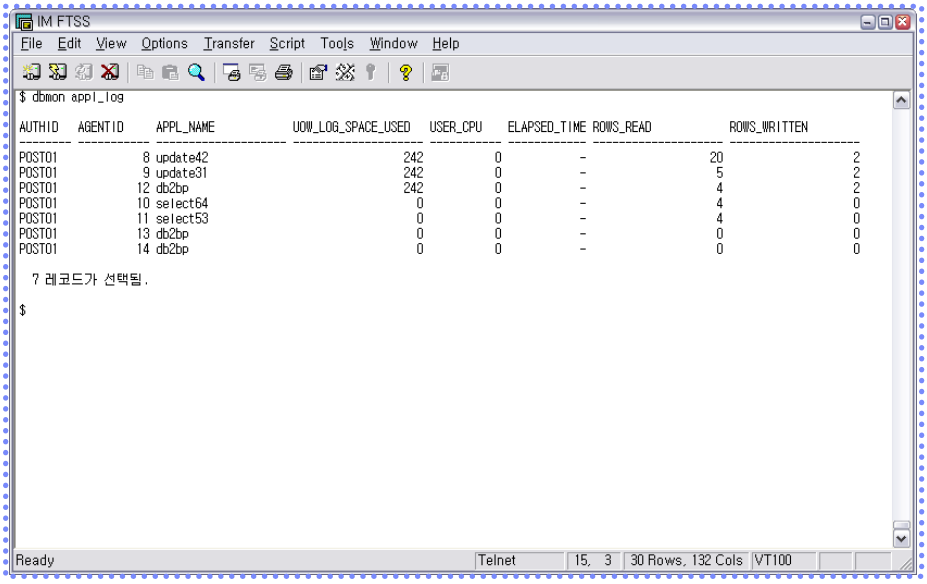


Figure 1417A ••• 응용프로그램별 로그 사용량

TOPIC 14-18 데이터베이스별 로그 사용량

Point

한 데이터베이스 전체 로그 사용량과 가장 오래된 트랜잭션을 가진 에이전트 ID 를 확인할 수 있습니다. 데이터베이스의 로그 사용량 비율이 심하게 높은 경우에는 가장 오래된 트랜잭션에 대한 점검이 필요합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2 를 사용합니다.

Tip

$MON_APPL_ID 는 스냅샷 함수를 이용한 SQL문을 실행하는 세션의 ID 입니다. 세션의 응용프로그램 ID 를 확인하여 모니터링의 결과에서 제외시킵니다. 포함시켜도 무방합니다.

1 아래의 SQL문을 실행합니다.

```
select
    total_log_used
    ,total_log_available
    ,tot_log_used_top
    ,cast(sec_logs_allocated as integer) sec_logs_allocated
    ,sec_log_used_top
    ,cast(appl_id_oldest_xact as integer) oldest_tx_agent
    ,substr(appl_info.appl_name,1,12) appl_name
from table(snapshot_database('$DBNAME', $DPMODE )) as database,
    table(snapshot_appl_info('$DBNAME', $DPMODE)) as appl_info
where database.appl_id_oldest_xact = appl_info.agent_id
and appl_info.appl_id <> '$MON_APPL_ID';
```

2 실행 결과는 다음과 같습니다.

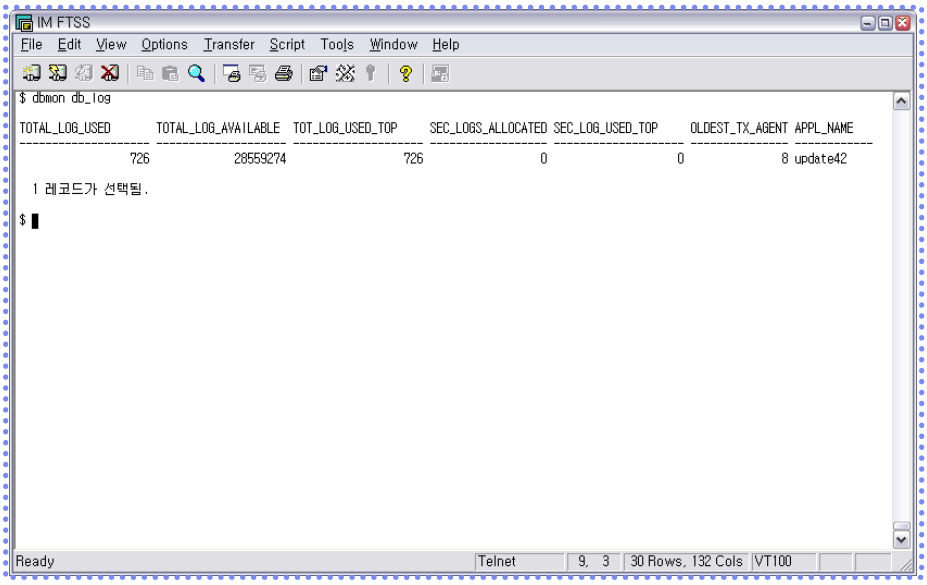


Figure 1418A ••• 데이터베이스별 로그 사용량

TOPIC 14-19 테이블 스페이스 사용량

Point

테이블 스페이스의 사용량과 사용율을 확인합니다. SMS 테이블스페이스는 필요할 때마다 공간을 할당하므로 거의 100%의 사용 비율을 보입니다. DMS 테이블스페이스의 경우에 70 ~ 80% 이상을 사용했다면, 컨테이너를 확장하는 등의 관리가 필요합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(tbs_cfg.tablespace_name,1,20) tablespace
    ,cast(tbs_cfg.tablespace_state as smallint) as state
    ,case tbs_cfg.tablespace_type
        when 1 then 'SMS'
        else 'DMS'
    end type
    ,cast(page_size as integer) page_size
    ,(total_pages * page_size) / 1024 / 1024 as total_size_mb
    ,(used_pages * page_size) / 1024 / 1024 as used_size_mb
    ,(free_pages * page_size) / 1024 / 1024 as free_size_mb
    ,case tablespace_type
        when 1 then 100
        else dec((tbs_cfg.used_pages * 100.00)/tbs_cfg.total_pages,5,2)
    end  as ratio
from table(snapshot_tbs_cfg('$DBNAME',$DPMODE)) as tbs_cfg
order by tablespace;
```

2 실행 결과는 다음과 같습니다.

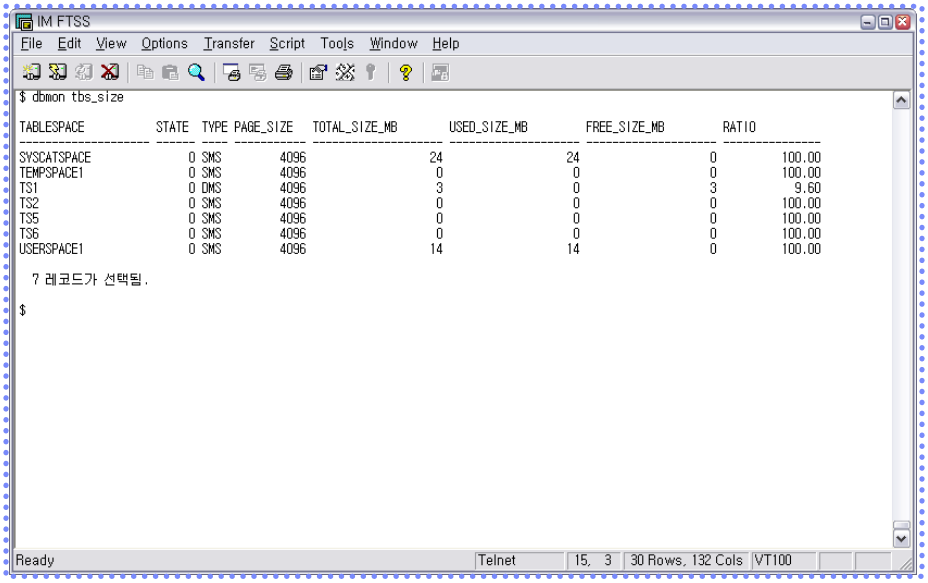


Figure 1419A ••• 테이블 스페이스 사용량

TOPIC 14-20 테이블 스페이스 컨테이너 사용량

Point

테이블스페이스의 각 컨테이너별 사용량을 확인합니다. 데이터는 기본적으로 라운드 로빈 방식으로 각 컨테이너에 균등하게 저장되므로, 한 테이블스페이스에 속한 각 컨테이너는 유사한 사용 비율을 보이게 됩니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select  distinct substr(container.tablespace_name,1,20) as tablespace
    ,substr(container.container_name,1,50) as container_name
    ,cast(container.total_pages as integer) as total_pages
    ,cast(container.usable_pages as integer) as usable_pages
    ,cast(container.total_pages*tbs_cfg.page_size/1024/1024 as int)
         as total_size_M
    ,cast(container.usable_pages*tbs_cfg.page_size/1024/1024 as int)
         as usable_pages_M
    ,case container.accessible
        when 1 then 'YES'
        when 0 then 'NO'
        else cast(container.accessible as char(1))
    end access
from    table(snapshot_container('$DBNAME', $DPMODE)) as container,
    table(snapshot_tbs_cfg('$DBNAME', $DPMODE)) as tbs_cfg
where container.tablespace_id = tbs_cfg.tablespace_id
order by tablespace, container_name;
```

2 실행 결과는 다음과 같습니다.

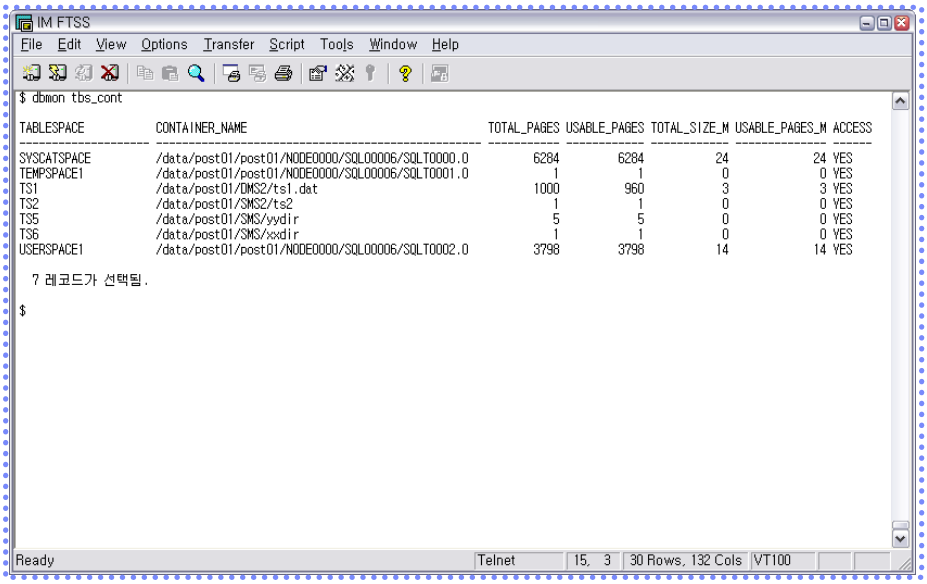


Figure 1420A ••• 테이블 스페이스 사용량

TOPIC 14-21 테이블 스페이스 적중률

Point

테이블스페이스의 데이터가 버퍼풀에 있는 비율을 확인합니다. 테이블의 데이터와 인덱스의 데이터에 대한 버퍼 적중율이 높을수록 유리합니다. 특정 테이블에 대한 적중률이 심하게 낮은 경우에는 버퍼풀의 크기를 조절하거나, 별도의 절 할당하도록 합니다.

Tip

$DBNAME은 모니터링의 대상인 데이터베이스명입니다. 기본 데이터베이스명인 SAMPLE로 설정되어 있으므로, 적절한 이름으로 변경합니다.

Tip

$DPMODE는 모니터링의 대상인 파티션 번호입니다. 파티션 번호를 지정하거나, 전체 파티션을 의미하는 -2를 사용합니다.

1 아래의 SQL문을 실행합니다.

```
select substr(tbs.tablespace_name,1,20) as tablespace
    ,cast(tbs.pool_data_l_reads as integer)  pool_data_l_reads
    ,cast(tbs.pool_data_p_reads as integer) pool_data_p_reads
    ,case tbs.pool_data_l_reads
        when 0 then null
        else dec(((tbs.pool_data_l_reads - tbs.pool_data_p_reads)
             * 100.00 / tbs.pool_data_l_reads) ,5,2)
    end data_hit_ratio
    ,cast(tbs.pool_index_l_reads as integer) pool_index_l_reads
    ,cast(tbs.pool_index_p_reads as integer) pool_index_p_reads
    ,case tbs.pool_index_l_reads
        when 0 then null
        else    dec(((tbs.pool_index_l_reads - tbs.pool_index_p_reads)
             * 100.00 / tbs.pool_index_l_reads),5,2)
    end index_hit_ratio
from table(snapshot_tbs('$DBNAME', $DPMODE)) as tbs
order by tablespace;
```

2 실행 결과는 다음과 같습니다.

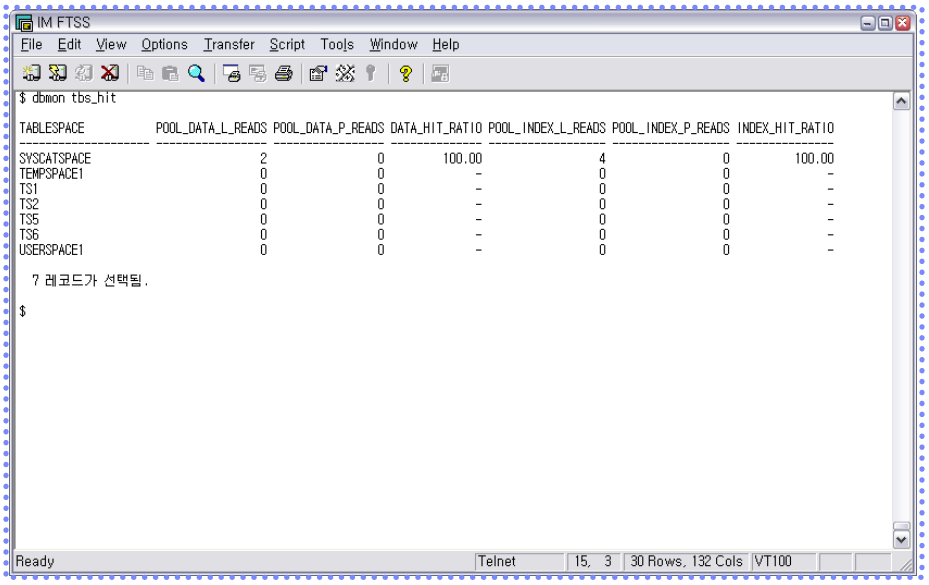


Figure 1421A • • • 테이블 스페이스 적중률

TOPIC 14-22 이벤트 모니터

Point

일정 기간 동안의 데이터베이스의 활동 내역에 대한 모니터링 정보를 수집하여 파일 또는 테이블에 저장하는 도구입니다. CREATE EVENT MONITOR 명령어, SET EVENT MONITOR 명령어, DROP EVENT MONITOR 명령어를 이용하여 관리합니다.

Tip

CONNECTIONS,STATEMENTS, TRANSACTIONS 유형은 수집되는 정보의 양이 너무 많게 되므로 모니터링의 범위를 줄일 수 있는 조건을 지정합니다.

Tip

테이블 이벤트 모니터를 사용할 때는 별도의 테이블스페이스를 생성하는 것이 좋습니다.

1. 특정한 이벤트가 발생할 때마다 이벤트 유형별로 관련된 이벤트 정보를 수집합니다. 이벤트의 유형은 다음과 같이 6가지로 분류됩니다.

이벤트 유형	설명
DATABASE	마지막 응용프로그램이 접속을 종료할 때, 데이터베이스에 관련된 모든 모니터링 항목을 기록합니다.
TABLES	마지막 응용프로그램이 접속을 종료할 때, 변경이 발생했던 테이블에 대해 읽혀진 행의 수와 기록된 행의 수 등을 기록합니다.
TABLESPACES	마지막 응용프로그램이 접속을 종료할 때, 각각의 테이블스페이스에 대해 버퍼풀 모니터링 항목과 프리페처, 페이지 클리너, 직접 I/O 회수 등의 정보를 기록합니다.
BUFFERPOOLS	마지막 응용프로그램이 접속을 종료할 때, 각각의 버퍼풀에 대한 기본 모니터링 항목과 프리페처, 페이지 클리너, 직접 I/O 회수 등의 정보를 기록합니다.
CONNECTIONS	응용프로그램이 접속을 종료할 때, 응용프로그램과 관련된 모든 모니터링 항목을 기록합니다.
STATEMENTS	SQL문이 실행을 종료할 때마다 SQL문의 시작과 종료 시간, 사용한 CPU, 동적 SQL문의 텍스트 등을 정보를 기록합니다.
DEADLOCKS	DB2 9.7부터 LOCKING 이벤트 모니터 사용을 권장합니다,
TRANSACTIONS	DB2 9.7부터 UOW 이벤트 모니텅 사용을 권장합니다.

2. 이벤트 모니터가 수집한 정보는 파일, 테이블, 파이프 등에 저장됩니다. 저장 방식에 따라 수집된 정보를 분석하는 방법이 다릅니다.

3. 특정 이벤트가 발생할 때마다 관련된 모니터링 정보를 수집하려면 데이터베이스에 접속한 후에 create event monitor 명령어를 이용하여 이벤트 모니터를 생성합니다.

4. set event monitor 명령어는 원하는 시점에 이벤트 모니터의 상태값을 ON 또는 OFF 로 설정하여 이벤트 모니터의 모니터링 기간을 조절할 수 있습니다. 상태값이 ON 이면 이벤트 모니터는 활성화되어 이벤트의 발생 정보를 수집합니다. 상태값이 OFF 이면 이벤트가 발생해도 이벤트 모니터는 정보를 수집하지 않습니다.

5. 이벤트 모니터를 제거하려면 drop event monitor 명령어를 이용합니다. 상태값인 ON 으로 설정된 이벤트 모니터는 제거할 수 없습니다.

6. 이벤트 모니터에 대한 정보는 SYSCAT.EVENTMONITORS 뷰를 이용해서 확인합니다.

```
$ db2 "select * from syscat.eventmonitors"
```

TOPIC 14-22 이벤트 모니터

Point

Deadlock 및 Transaction에 대한 이벤트 모니터가 9.7 이후 Locking 및 UOW로 기능 대체됩니다.

7 Deadlock 모니터링 방법입니다.

9.7 이전)
CREATE EVENT MONITOR FOR DEADLOCKS
CREATE EVENT MONITOR FOR DB2DETAILDEADLOCK
9.7 이후)
CREATE EVENT MONITOR FOR LOCKING

8 트랜잭션 모니터링 방법입니다.

9.7 이전)
CREATE EVENT MONITOR FOR TRANSACTIONS
9.7 이후)
CREATE EVENT MONITOR FOR UNIT OF WORK

TOPIC 14.23 CREATE EVENT MONITOR 문

Point

특정한 데이터베이스에 이벤트 모니터를 생성하는 SQL문입니다. 8가지 이벤트 유형을 선택적으로 지정할 수 있고, 수집된 정보를 저장할 파일, 테이블, 파이프의 정보를 지정합니다.

Tip

SYSADM 또는 DBADM 권한이 필요합니다.

1 create event monitor 의 형식은 다음과 같습니다.

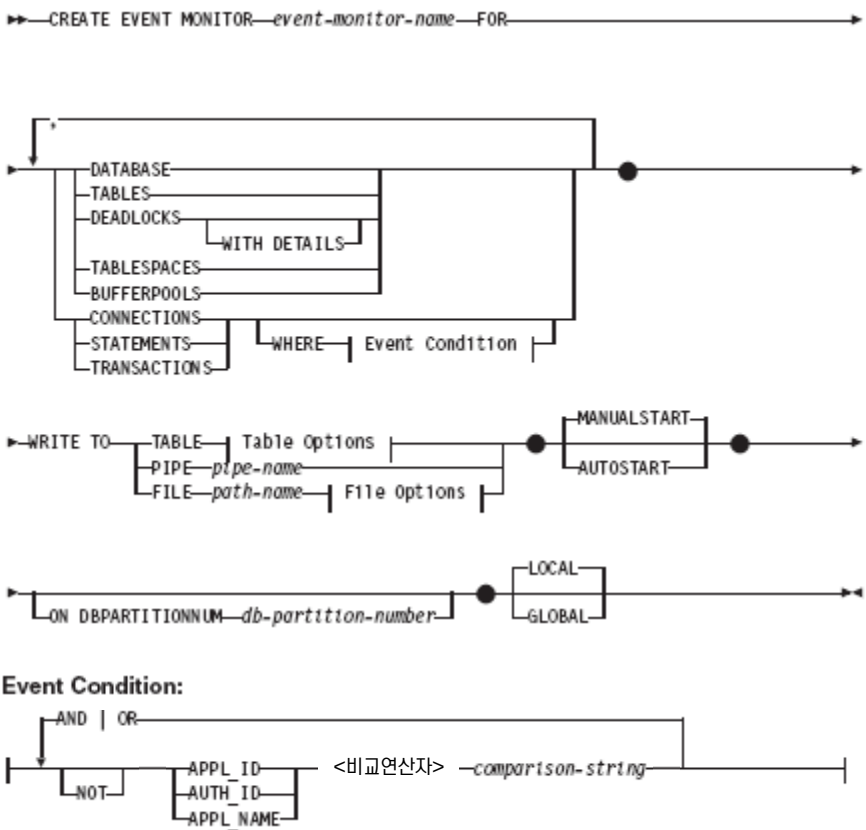


Figure 1423A ··· CREATE EVENT MONITOR 문

2 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
<이벤트모니터명>	임의의 고유한 값으로 지정합니다.
FOR <이벤트유형>	8가지 유형 중에서 한 개 이상을 선택합니다.
WHERE	CONNECTIONS, STATEMENTS, TRANSACTIONS 유형은 수집되는 정보의 양이 너무 많게 되므로 모니터링의 범위를 줄일 수 있는 조건을 지정합니다. APPL_ID, AUTH_ID, APPL_NAME 을 이용하여 조건식을 표현합니다.
WRITE TO TABLE	수집된 정보를 테이블에 저장합니다.
WRITE TO FILE	수집된 정보를 파일에 저장합니다.
WRITE TO PIPE	수집된 정보를 파이프에 저장합니다.
MANUALSTART	데이터베이스가 활성화되어도 자동적으로 시작되지 않습니다. SET EVENT MONITOR STATE 문으로 조절합니다.
AUTOSTART	데이터베이스가 활성화되면 자동적으로 시작됩니다.

TOPIC 14-24 파일 이벤트 모니터

Point

이벤트 모니터가 수집한 정보를 파일에 저장합니다. 파일이 생성될 디렉토리, 파일의 크기, 파일의 개수, 버퍼의 크기 등의 옵션을 추가적으로 지정할 수 있습니다. 생성된 파일은 db2evmon 명령어로 변환하여야 합니다.

1 create event monitor 의 형식은 다음과 같습니다.

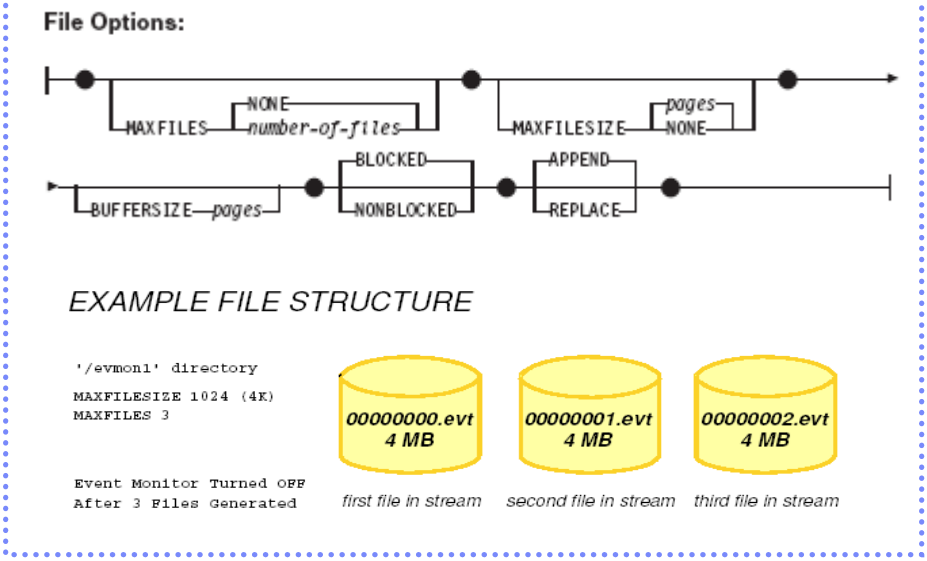


Figure 1424A • • • 파일 이벤트 모니터의 옵션

Tip

수집된 정보는 출력용 디렉토리에 *.evt 파일에 저장됩니다.

2 이벤트 모니터가 수집하는 정보를 저장할 출력용 디렉토리를 생성합니다.

```
$ mkdir –p <출력용 디렉토리>
```

Tip

출력 파일의 개수가 MAXFILES 옵션의 값을 초과하거나, 데이터베이스가 비활성화되면 이벤트 모니터는 자동적으로 중지됩니다.

3 create event monitor 명령어로 파일 이벤트 모니터를 생성합니다.

```
$ db2 “create event monitor <이벤트모니터명> for <이벤트유형명> write to
    file <출력용 디렉토리명> MAXFILES <최대 파일 개수> MANUALSTART”
```

4 이벤트 모니터를 활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 1”
```

5 일정한 시간동안 이벤트 정보를 수집한 후에 이벤트 모니터를 비활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 0”
```

Tip

파일 이벤트 모니터의 출력 파일은 db2evmon 명령어로 분석합니다. 이벤트 모니터가 생성한 출력 파일인 *.evt 는 제거해도 됩니다.

6 db2evmon 명령어를 이용하여 파일로 수집된 이벤트 정보를 텍스트 형태로 변환합니다.

```
$ db2evmon -db <데이터베이스명> -evm <이벤트모니터명> > <출력파일명>
$ rm -Rf  <출력용 디렉토리>/*.evt
```

7 불필요한 이벤트 모니터는 제거합니다.

```
$ db2 drop event monitor <이벤트모니터명>
```

TOPIC 14-25 테이블 이벤트 모니터

Point

이벤트 모니터의 결과를 테이블에 저장합니다. 수집된 데이터 스트림을 한 개 이상의 논리적인 그룹으로 분할하여 해당 테이블에 INSERT 합니다. SELECT 문을 이용하여 분석합니다.

Tip

테이블 이벤트 모니터를 사용할 때는 별도의 테이블스페이스를 생성하는 것이 좋습니다.

Tip

테이블 이벤트 모니터의 정보는 테이블에 저장되므로, 일반 테이블과 동일한 방법으로 조회하면 됩니다.

Tip

pctdeactivate 옵션은 DMS 테이블스페이스에만 적용되며, 지정한 비율 이상을 사용하게 되면 이벤트 모니터를 중지합니다.

1 옵션은 다음과 같습니다.

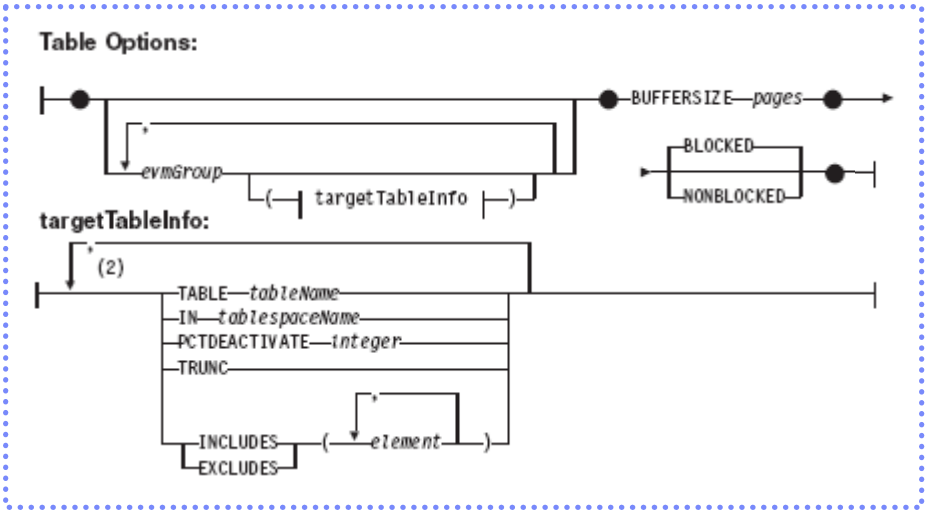


Figure 1425A ••• 파일 이벤트 모니터의 옵션

2 이벤트 모니터가 수집할 정보를 저장할 테이블스페이스를 생성합니다.

```
db2 "create tablespace <TS명> managed by database using (...)"
```

3 이벤트 모니터를 생성합니다.

```
db2 +p -t << EOF
    create event monitor <이벤트모니터명> for <이벤트유형>
    write to table
    connheader (table mon2.connheader, in <TS명>,
                pctdeactivate 90),
    stmt (table mon2.statements,  in <TS명>,  pctdeactivate 90),
     control (table mon2.control, in <TS명>, pctdeactivate 90)
    manualstart;
EOF
```

4 이벤트 모니터를 활성화시킵니다. 일정한 시간 동안 이벤트 정보를 수집한 후에 이벤트 모니터를 비활성화시킵니다.

```
$ db2 “set event monitor <이벤트모니터명> state = 1”
$ db2 “set event monitor <이벤트모니터명> state = 0”
```

5 불필요한 이벤트 모니터는 제거합니다.

```
$ db2 drop event monitor <이벤트모니터명>
```

TOPIC 14-26 시간 소요 모니터

Point

포괄적인 시간 기반 모니터 요소 세트를 사용하면 어디에, 어떻게 시간이 사용되었는지를 쉽게 이해할 수 있습니다.

Tip

- 정밀한 성능 튜닝을 위해 9.7에서 시간 기반 모니터링이 개선되었습니다.

1. 포괄적인 시간 소요 모니터링 요소를 통하여, 시간이 어디에 소요되는지를 정확하게 표시하는 기능으로 문제점의 잠재적인 원인을 쉽게 찾고 성능 개선을 위해 조정을 수행할 수 있을지 여부를 확인할 수 있습니다.

전체 시간 소요 처리 요청 및 DB2 데이터베이스 관리 프로그램 내의 전체 대기 시간
자원(예: 잠금, 버퍼 풀 또는 로깅)에 의한 대기 시간
DB2 데이터베이스 관리 프로그램 외부의 시간 소요 측정(client_idle_wait_time)

2. 시간 모니터링 결과값을 수집하여 아래와 같은 도표를 산출할 수 있습니다. 아래 예제는 전체 대기 시간 중 Lock 대기 시간이 가장 큰 부분을 차지함을 알 수 있습니다.

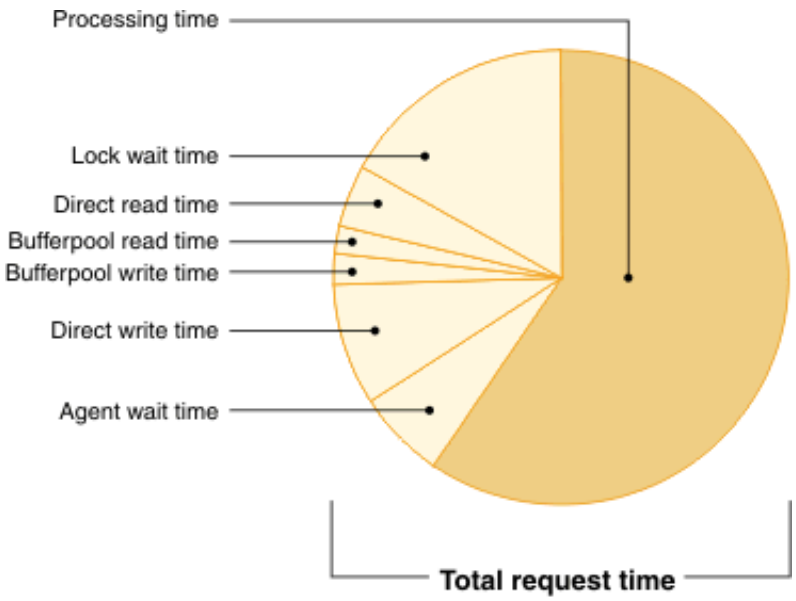


Figure 1426A • • • 시스템 전체 요청시간 예제

TOPIC 14-27 DB2PD 모니터링

Point

DB2PD는 메모리로부터 수집되어 스냅샷 모니터에 비해 보다 가볍고 신속한 데이터베이스 모니터을 제공합니다.

Tip

Db2pd는 메모리 세트의 빠르고 즉각적인 정보를 포함하는 문제점 판별 도구입니다. 반면에 스냅샷 모니터는 래치를 확보하거나 엔진자원을 사용합니다.

1 db2pd 로 데이터베이스 시스템 메모리 세트에서 정보를 검색합니다.

Figure 1427A ··· DB2PD 사용법

옵션	설명
-inst	모든 인스턴스 레벨의 정보를 보여줍니다.
-h | help	온라인 도움말을 보여줍니다.
-v | -version	설치된 DB2제품의 현재버전과 서비스레벨을 보여줍니다.
-database | -db<DB명>	지정된 데이타베이스의 메모리집합에 접근하도록 지정합니다.
-alldatabases | -alldbsw	모든 데이타베이스 메모리 집합에 접근하도록 지정합니다.
-everything	모든 데이타베이스의 모든 옵션을 수행합니다.
-file <파일명>	결과값을 지정된 파일에 기록합니다.
-applications	애플리케이션에 대한 정보를 보여줍니다. 애플리케이션 ID가 지정될 경우 그 애플리케이션 정보만을 보여줍니다.
-appinfo	현재 UOW의 Dynamic SQL 구문의 실행을 포함하여 애플리케이션에 대한 자세한 정보를 제공합니다.

Figure 1427B ··· DB2PD 옵션 설명

TOPIC 14-28 db2top

Point

db2top은 db2 데이터베이스의 포괄적인 모니터링을 위해 기본으로 제공되는 도구입니다.

Tip

- db2top은 별도 제공되었으나, 9.7 이후 설치 시 기본 제공됩니다.

Tip

- 구체적인 사용 설명은 http://www.ibm.com/developerworks/data/library/techarticle/dm-0812wang/을 참조하십시오.

1 인스턴스 사용자로 로그인 하여 db2top 을 입력합니다.

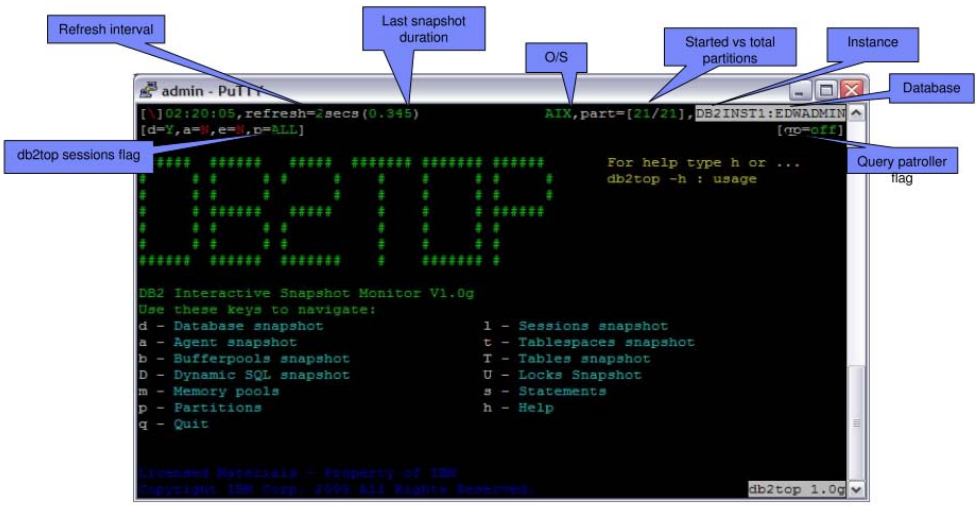


Figure 1428A • • • db2top 초기화면

2 db2top의 옵션은 아래와 같습니다. db2top의 어떠한 화면에서도 h를 누르게 되면 도움말이 나타납니다.

d - Database	l - Sessions
a - Details for agent <agentid>	t - Tablespaces
b - Bufferpools	
T - Tables	
D - Dynamic SQL	U - Locks
m - Memory pools	s - Statements
u - Utilities	p - Partitions
C - Toggle collector on/off	W - Watch user/agent
/ - Set regexp	
g - Toggle graph on/off	
i - Toggle idle objects on/off	G - Toggle local/global sn
P - Select db partition	X - Toggle extended mod
k - Toggle actual/delta values	z - Descending sort
Z - Ascending sort	+ - Longer default sort
- - Shorter default sort	I - Set new snapshot inter
R - Reset snapshot monitor	S - Run native DB2 snap
> - Move right	< - Move left
c - Change columns order	f - Freeze display
! - Goto to system prompt	V - Set default explain sc
O - Display settings	w - Write parms to .db2t
h - Help	q - Quit

TOPIC 14-29 db2top - Application

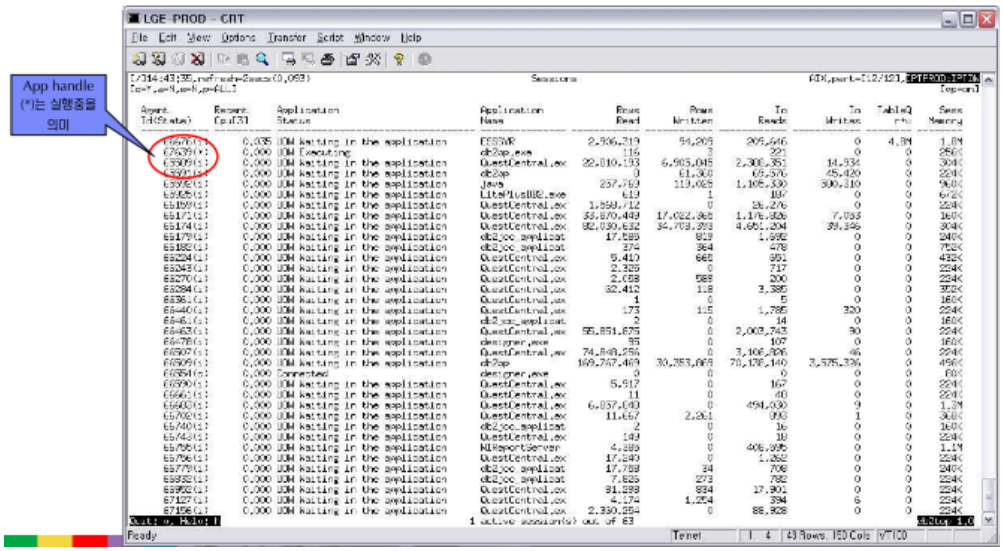
Point

db2top을 이용한 응용프로그램 모니터링 수행 방법입니다.

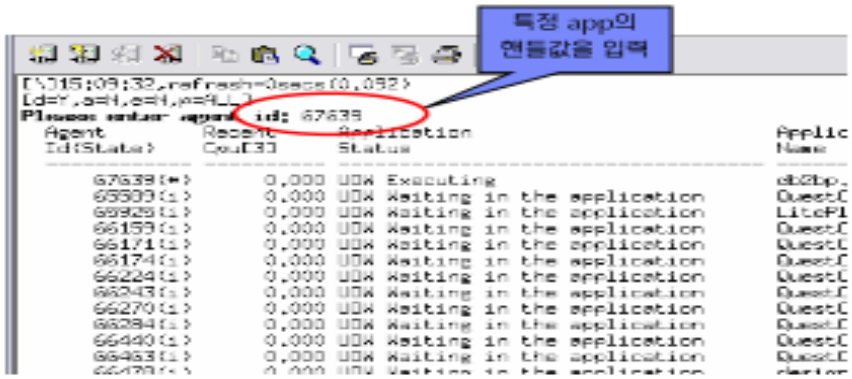
Tip

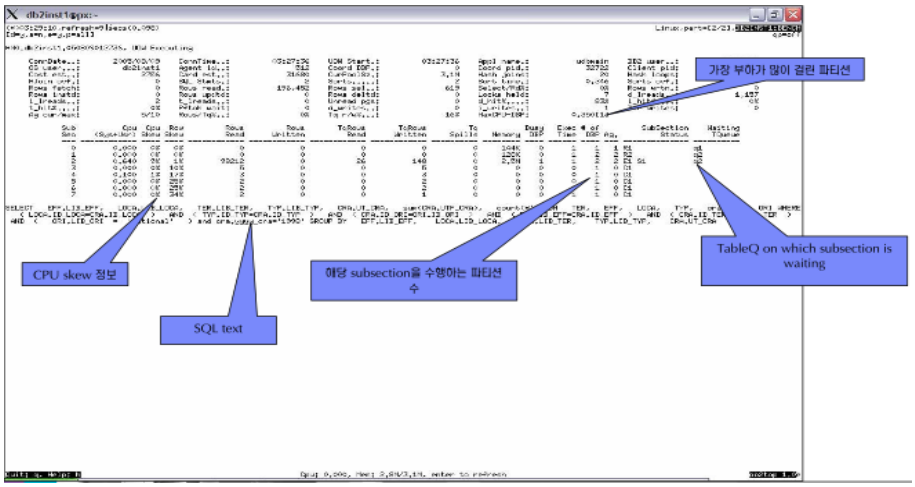
- f 키를 사용하면 특정 Appliction을 작업 중지시킬 수 있습니다.

1 l 키를 사용하여 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2 a 키를 사용하여 특정 application의 핸들값을 입력하면 자세한 정보를 확인 할 수 있습니다.





TOPIC 14-29 db2top - Application

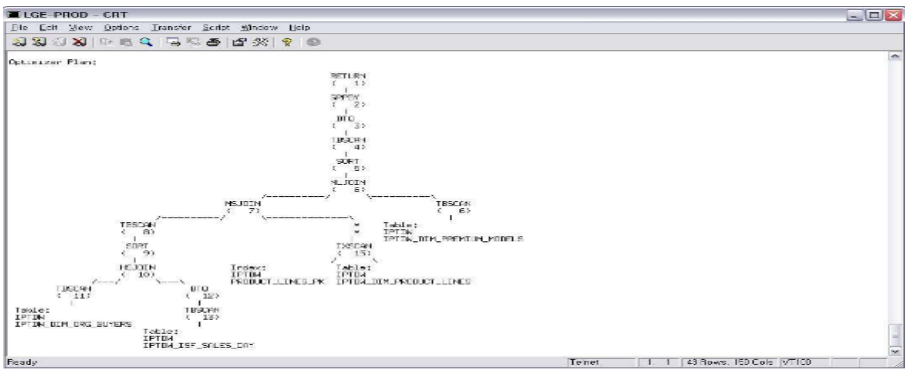
Point

db2top을 이용한 세부 응용프로그램 모니터링 수행 방법입니다.

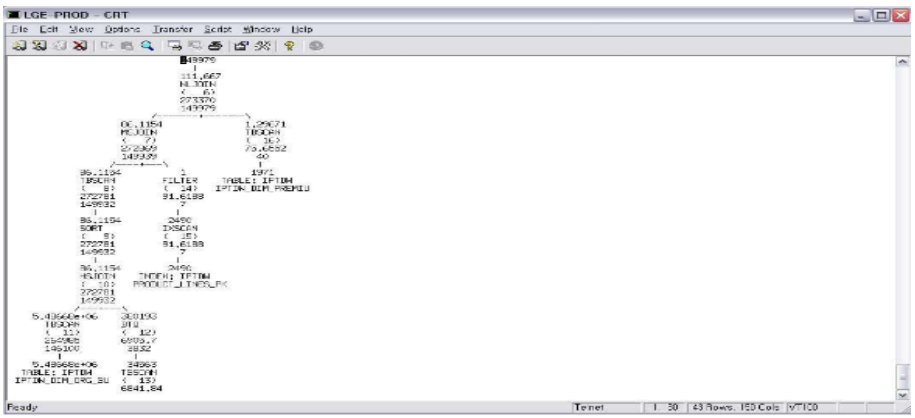
Tip

vi editor를 통해 처리되기 때문에 :q를 통해 완료됩니다.

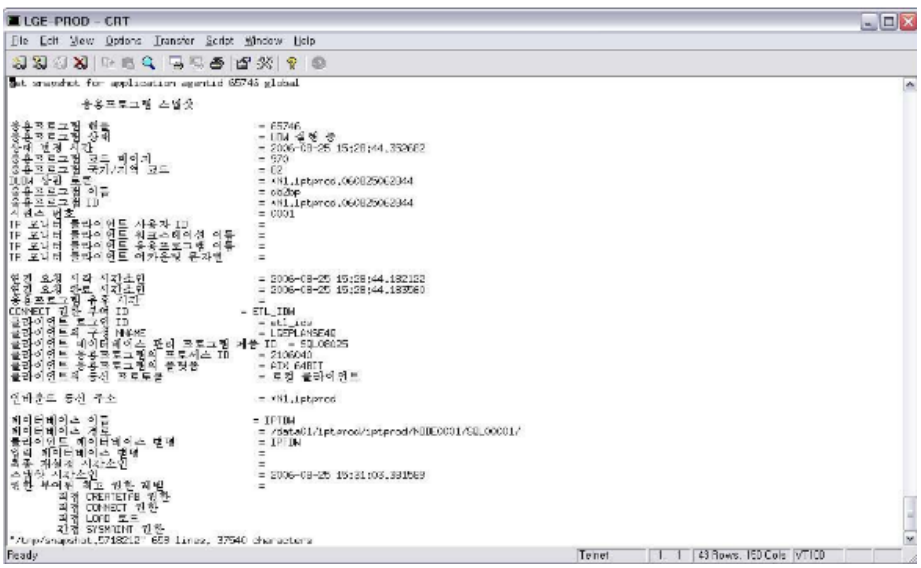
3 e 키를 사용하면 db2expln을 사용하여 해당 app의 access plan을 검증할 수 있습니다. vi editor를 통해 처리되기 때문에 :q를 통해 완료됩니다.



4 x 키를 사용하면 db2exfmt를 사용하여 해당 app의 access plan을 검증할 수 있습니다.



5 s 키는 해당 app에 대한 db2 snapshot을 수행할 수 있습니다. 이를 통해 보다 자세한 데이터를 수집하고 모니터링 할 수 있습니다.

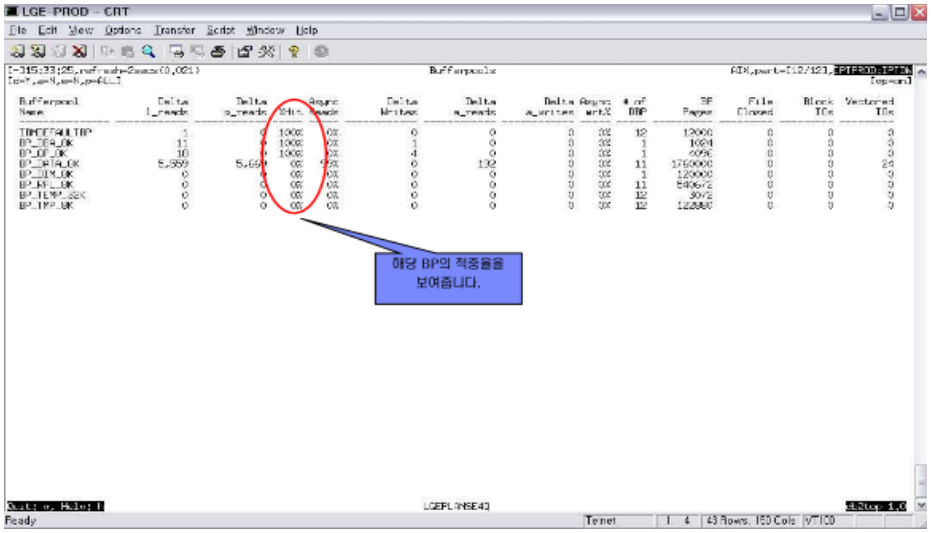


TOPIC 14-30 db2top - Memory

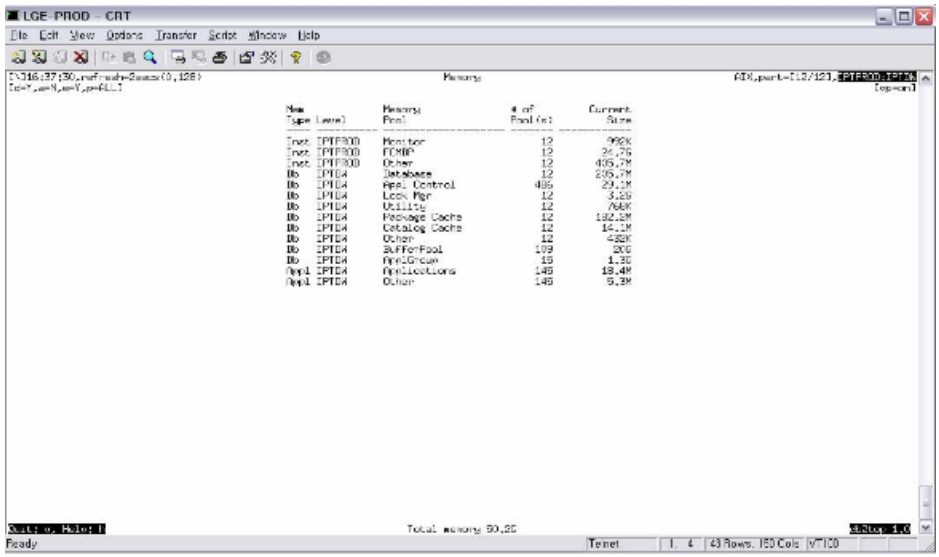
Point

db2top을 이용한 메모리 모니터링 수행 방법입니다.

1. b 키를 사용하여 bufferpool을 모니터링 할 수 있습니다.
좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2. m 키를 사용하여 DB2가 사용하는 전체 메모리를 모니터링 할 수 있습니다.

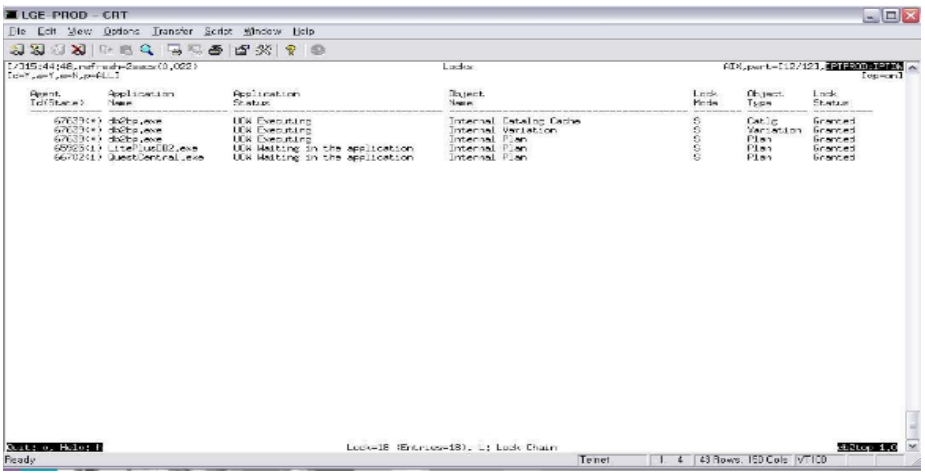


TOPIC 14-31 db2top – Lock

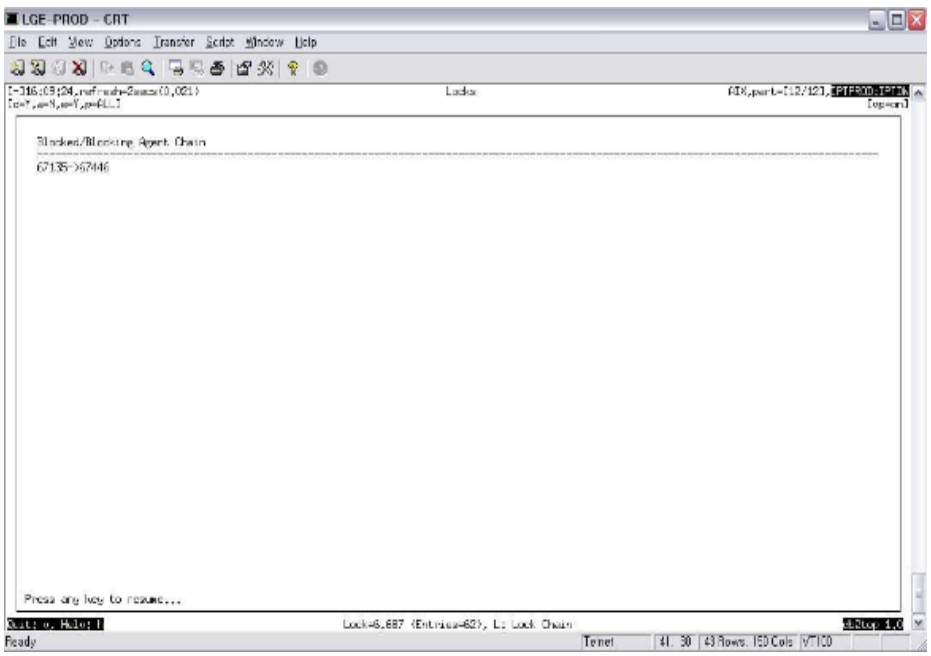
Point

db2top을 이용한 Lock 모니터링 수행 방법입니다.

1 U 키를 사용하여 lock을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다



2 L 키를 사용하여 lock chain을 확인할 수 있습니다.



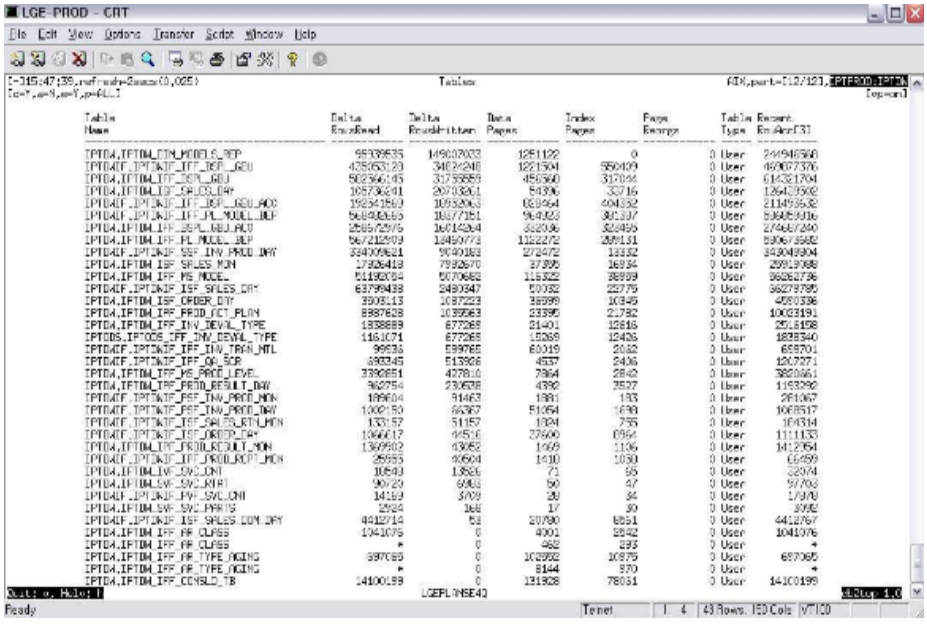
TOPIC 14-32 db2top – Table

Point

db2top을 이용한 테이블 모니터링 수행 방법입니다.

1 T 키를 사용하여 테이블을 모니터링 할 수 있습니다.

좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

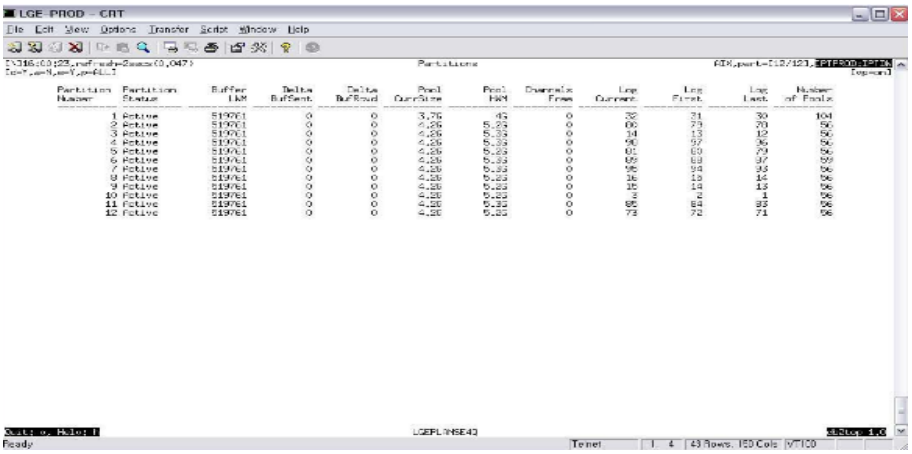


TOPIC 14-33 db2top – Partitioning

Point

db2top을 이용한 파티셔닝 모니터링 수행 방법입니다.

1 p 키를 사용하여 partition을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

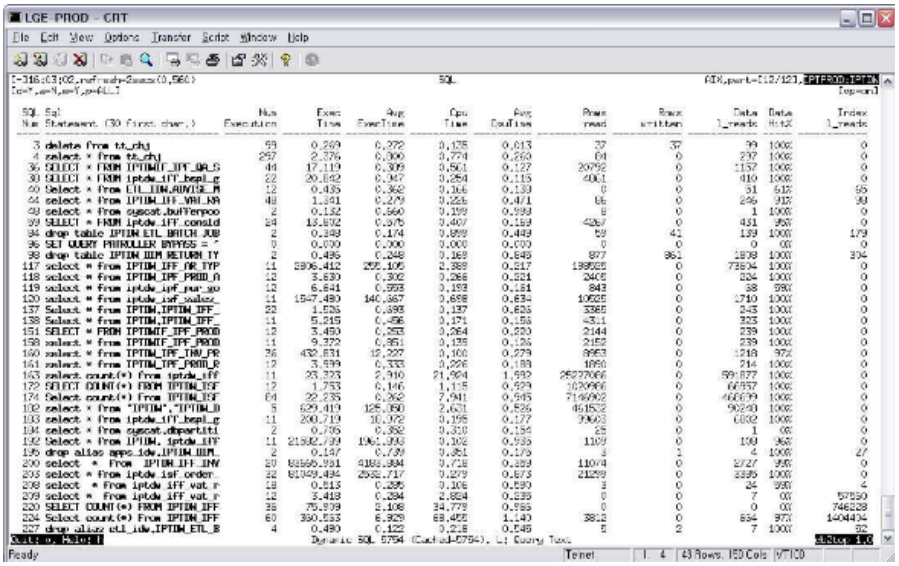


TOPIC 14-34 db2top – Dynamic SQL

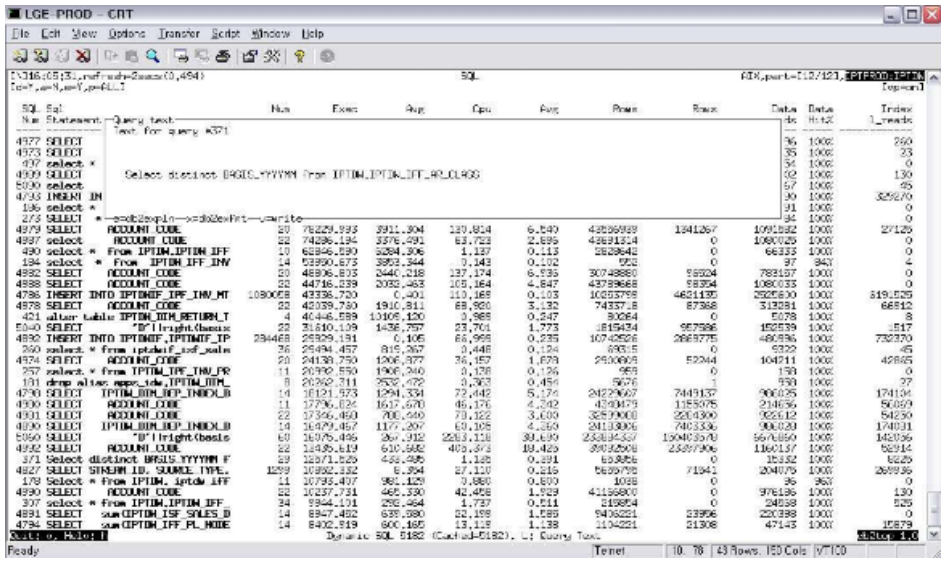
Point

db2top을 이용한 파티셔닝 모니터링 수행 방법입니다.

1 D 키를 사용하여 Dynamic SQL을 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.



2 L 키를 사용하여 SQL 전체 문장을 확인할 수 있습니다. w 키를 사용하여 SQL을 파일로 저장할 수 있습니다. 파일은 홈디렉토리에 dynsql.sql라는 이름으로 생성됩니다.

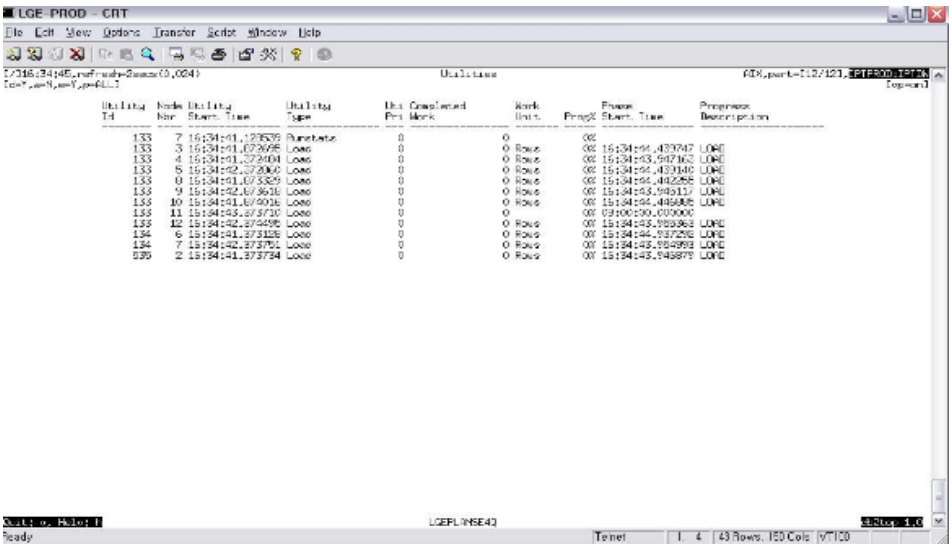


TOPIC 14-35 db2top – Utility

Point

db2top을 이용한 유틸리티 모니터링 수행 방법입니다.

1 u 키를 사용하여 Utility를 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다.

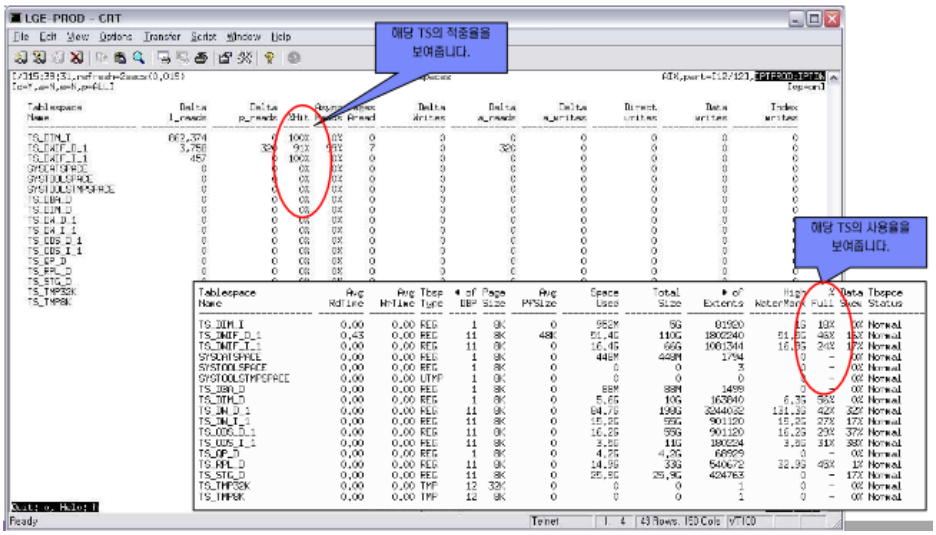


TOPIC 14-36 db2top - Tablespace

Point

db2top을 이용한 Tablespace 모니터링 수행 방법입니다.

1 t 키를 사용하여 tablespace를 모니터링 할 수 있습니다. 좌우 화살표(← →)를 이용하여 모니터링 내용을 이동하며 확인할 수 있습니다



Memo