Goodus 기술노트 [41 회]

Diagnostic Events

Author	노재구, 유현재
Creation Date	**2009-05-26**
Last Updated	**2009-06-07**
Version	**1.1**

Version	변경일자	변경자(작성자)	주요내용
1.1	2009-06-07	노재구	문서 보완

Goodus
Breathing Life into Technology

Contents

1. Diagnostic Events

1.1. Event 사용 목적

DBA 가 오라클 DB 을 관리 중 DBMS 의 상태나 장애 현상이 발생했을 때 다양한 로그을 통해 정보를 제공 받을 수 있다. 기본적인 로그 파일에서는 DB 가 Crash 되거나 Hang 이나 Loop 상태에 빠졌을 때, 해당 장애 상황을 판단하거나 분석하는 것이 어렵다. 현 장애 상황이 도대체 어떤 원인에 의해 발생했으며, 어떤식으로 분석해야 할지는 DBA 의 몫일 것이다. DBA 는 불충분한 정보를 추가적으로 얻어내어 그 결과를 오라클 SR 을 통해서 정확한 분석을 의뢰할 수 있다.

오라클에는 대기 이벤트와 진단 이벤트로 두가지 타입의 Event 를 제공한다. 여기서의 Event 는 오라클 내부에서 일어 날 수 있는 사건을 의미한다. 대기 이벤트란 어떤 사건이 종료되기를 기다릴 때 사용되는 이벤트로 인덱스 스캔시 대기한 이벤트 'db file sequential read' 를 예를 들 수 있다. 그리고 진단 이벤트란 어떤 사건이 발생했을 때 진단을 수행하는 목적으로 사용되는 이벤트이다. 여기서는 진단 이벤트를 주제로 다룰 것이다.

1.2. Event 분류

진단 이벤트는 다음과 같은 네가지의 카테고리로 분류할 수 있다.

- Immediate Dump Events : 트레이스 파일에 수집한 정보를 내보낸다. File headers(control file, redo log files, data files), System state dumps, Process state dump 가 이에 해당한다.
- On-Error Dump Events : immediate dumps 와 유사하지만, 에러가 발생했을 때만 제공된다. 가장 일반적인 예가 Error Stack 이다. 장애시 process call stack 및 기타 다른 정보를 제공한다.
- Change Behavior Events : 이 이벤트를 사용하여, 인위적으로 오라클 서버를 작동을 제어하거나, 숨겨진 기능을 활성화하거나 문제의 해결책으로서 사용하거나, 특별한 튜닝을 수행할 목적으로 사용할 수 있다.
- Trace Events : 실행 중인 Processes 에 대한 트레이스를 수행한다. 이 트레이스는 문제를 이해하고 분석하고 해결하는데 사용된다.

1.3. Event 설정 구문 명세

```
<event name> <action> {:<event name> <action>}*
<action>::= "trace" <action qualifier>{,<action qualifier>}*
<action qualifier>::= <trace spec> {;<trace spec>}*
<trace spec>::= "name" <trace name> <trace qualifier> {, <trace qualifier>}*
<trace qualifier>::= ""|"off"|"after <n> times"|"forever"|"lifetime <n>"|"level <n>"
```

- <event name> : Event 와 관련된 상징적인 이름, 또는 이벤트 번호로 부여된다. 이벤트명을 지정하면, Parser 는 Event Name Table 로 부터 찾고, 해당 이벤트명 지정하지 않으면 "Immediate"로 설정한다. "Immediate" 는 특별한 이벤트명으로 즉각적이고 무조건적인 인벤트로 지칭되며, 설정 이후 기다림 없이 바로 수행된다. 그러므로 Immediate 이벤트는 파라메터 파일로 세팅할 수 없다.
  <event name> 이 번호이면, Oracle Error Number(OER) 이거나, internal 코드이다. Internal 코드는 10000 ~ 10999 사이의 값을 취할 수 있다.

Unix 인 경우 에러코드와 Internal 코드는 "$ORACLE_HOME/rdbms/mesg/oraus.msg " 메시지 파일에 정리되어 있다. VMS 는 ORA_RDBMS 에 설명되어 있으며, Windows 인 경우에서는 제공되는 파일이 없다.

- **<action>** : event 가 발행했을 시의 어떠한 행위(action) 및 행위에 대한 레벨을 설정한다. 일반적은 구문 형식은 아래와 같다.

  <event name> "trace" "name" <trace name> <trace qualifier> {, <trace qualifier>}*

- **<trace name>** : 문맥에 독립적인 debug dump operation 과 관련된 트레이스에 대한 상징적인 이름이다. 이벤트명이 "**immediate**" 인 경우 특별하게 **level<n>** 을 제외하고 어떠한 <trace qualifier>를 허락하지 않는다. "**context**" 이벤트명은 특정 문맥에 관련한 트레이스를 수행함을 의미하며, 그로 인해 연관된 debug dump operation 을 야기하지는 않는다. 그 대신에 이벤트를 수행한 호출자를 리턴하고, 그 이벤트에 대해 context tracing 이 활성화(active) 상태인지를 알려주되, 활성화되어 있다면 trace level 이 무엇인지 알려준다. Trace level 은 dump 의 세부적인 레벨을 조정한다. Level number 를 증가시킴으로써 좀 더 세부적인 레벨을 설정할 수 있다. Level 은 최소값을 1 로 설정할 수 있으며, 각 trace 별로 설정 가능한 Level Number 가 다르다.
- **<trace qualifier>** : trace 를 한정하는 trace qualifier 는 다음과 같다.

Trace qualifier	Description
"" (nothing)	지정하지 않는 경우는 모든 기본 값이 사용된다.
"off"	현재 Event에 대해 trace를 비활성화시킨다
"after <n> times"	이 Event가 <n>번 일어난 후에 트레이스를 수행을 시작한다.
"forever"	한번 활성화되면, 이 이벤트가 매번 일어날 때 마다 트레이스가 수행된다.
"lifetime <n>"	한번 활성화되면, 이 이벤트가 <n>연속적으로 일어 났을 때 트레이스가 수행되고, 그 이후에 이 이벤트에 대한 트레이스는 비활성화된다.
"level <n>"	트레이스가 최초 활성화될 때, level 을 <n>으로 세팅한다.

표 1 - Trace Qualifiers

- 이벤트 분류별 구문에 대한 요약 표이다.

Uses of diagnostic dumps and events	Event Name	Trace Name
Immediate dump	IMMEDIATE	<dump>
On error dump	Error number	ERRORSTACK
Change behavior	Event number	CONTEXT
Trace	Event number	CONTEXT

표 2 - Event 구문 요약표

1.4. Event 세팅 방법

이벤트는 다음과 같이 세팅할 수 있다.

- Initialization parameter 에 EVENT 파리메터를 통해 세팅.

  event = "<event name> <action> {:<event name> <action>}*"

  예 1) 하나의 EVENT 파라메터를 설정한다.

  event = "604 trace name errorstack"

예 2) 두 개 이상의 EVENT 를 설정할 때는 두가지 방법으로 설정할 수 있다.

(1) 라인별로 나열하는 방법

```
event = "10015 trace name context forever"
event = "10046 trace name context forever, level 4"
```

(2) 콜론(:) 을 구분자로 나열하는 방법

```
event = "10015 trace name context forever:
          10046 trace name context forever, level 4"
```

예 3) 하나의 EVENT 에서 여러개의 ACTION 을 설정할 때는 세미콜론(;)을 사용한다.

```
event = "4031 trace name HEAPDUMP level 1 ; name ERRORSTACK level 3"
```

예 4) 하나의 EVENT 에 대해 여러개의 ACTION 을 설정할 때 콜론(:)을 사용할 경우 다음와 같은 구문에러가 발생한다.

```
event ="4031 trace name HEAPDUMP level 1:name ERRORSTACK level 3"

SQL> startup
ORA-02194: event specification syntax error 230 (minor error 210) near 'NAME
```

- ALTER {session | system } SET EVENTS 명령어로 세팅.

현재 세션에 대해 :

```
alter session set events '<event name> <action> {:<event name> <action>}*'
```

예) SQL> alter session set events '604 trace name errorstack';

모든 신규 세션에 대해:

```
alter system set events '<event name> <action> {:<event name> <action>}*'
```

예) SQL> alter system set events '604 trace name errorstack';

하나의 EVENT 에 대해 여래개의 ACTION 을 설정할 떄는 세미콜론을 사용한다.

예) SQL> alter session set event '4031 trace name HEAPDUMP level 1 ; name ERRORSTACK level 3';

하나의 EVENT 에 대해 여래개의 ACTION 을 설정할 떄 콜론을 사용하면 다음과 같은 구문에러가 발생한다.

```
예) SQL> alter session set events '4031 trace name
      2   HEAPDUMP level 1: name ERRORSTACK level 3';

    ORA-02194: event specification syntax error 230 (minor error 210) near 'NAME'
```

- sys.dbms_system.set_ev(..) procedure 를 통해 세팅.

다른 세션에 대해 : DBMS_SYSTEM 은 오로지 SYS 유저만이 실행권한을 가지는데 필요에 다라 다른 유저에게 이 패키지의 EXCUTE 권한을 부여해서 사용 가능하게 할 수 있다.

```
sys.dbms_system.set_ev(sid,serial#,<event>,<level>,'<action>')
예)  SQL> execute dbms_system.set_ev(8,219,10046,12,'');
```

- ORADEBUG 유틸리티 사용.

ORADEBUG 는 SQL*Plus 내에서만 사용가능 하며, ORADEBUG 를 사용하기 위해서는 반드시 SYSDBA 권한을 가진 유저로 접속해야 한다. "ORADEBUG help" 는 모든 명령어를 볼 수 있다. 단, WINDOWS Platform 은 지원하지 않는다.
한편, ORADEBUG 에서 process states 또는 system states dump 를 수행할 때에는 level 키워드는 사용되지 않는다.

```
oradebug <command>
```

예) 백그라운드 프로세스의 pid 를 알아낸다.

```
$  ps -ef | grep smon

oracle    30521       1   0 May28 ?          00:00:13 ora_smon_PROD

SQL> oradebug setospid 30521
Oracle pid: 8, Unix process pid: 30521, image: oracle@linux3 (SMON)

SQL> oradebug unlimit
Statement processed.

SQL> oradebug dump processstate 10
Statement processed.
```

1.5. Event 분류별 사용법

1.5.1. Immediate Dump Events

◆ Dump 를 수행하는 방법은 아래와 같다.

```
# alter system 또는 alter session 명령어 이용
alter session set events  'immediate trace name <dump> level <n>' ;

# ORADEBUG 사용
oradebug dump <dump> <level>

# DBMS_SYSTEM Package 사용
sys.dbms_system.set_ev(sid, serial#, 65535, <level>,  '<dump>' );
```

<dump> : immediate dumps - file_hdrs, redohdr, controlf, processstate, systemstate 등.
<level> : 각 덤프별로 제공되는 결과의 타입과 양을 결정한다.

예1) ORADEBUG 사용시 Control File Dump 수행

```
SQL> oradebug setospid 5681
SQL> oradebug dump controlf 10
```

예 2) DBMS_SYSTEM.SET_EV() 사용시 Control File Dump 수행

```
SQL> execute sys.dbms_system.set_ev(18, 120, 65536, 10, 'controlf');
```

Event Type	Dump	level	Comments
Files	CONTROLF	10	Dump Control file contents
	FILE_HDRS	10	Database File headers
	REDOHDR	10	On line log headers
Data Blocks	BLOCKDUMP	decimal DBA	Dump block @DBA (File ONLINE) (Oracle7 only)
	TREEDUMP	SegHDR DBA+1 (obj# in O8)	Index Tree dump
	SET_TSN_P1	ts#+1	O8 set TS# for BUFFER dumps
	BUFFER	DBA	Buffer Cache copies of DBA
	BUFFERS	1-10	Dump entire Buffer Cache (big)
Process	ERRORSTACK	0=Err Stk 1+Call Stk 2+Proc state 3+Context	level 3 most useful
	PROCESSSTATE	2+inc child	Use ERRORSTACK level 3 instead
	CONTEXTAREA	2+inc child	Use ERRORSTACK level 3 instead
	HEAPDUMP	Version Dependent	Dump PGA,UGA or SGA etc
	HEAPDUMP_ADDR	ADDRESS	oracle8 Dump heap at ADDRESS
	SAVEPOINTS	ADDRESS >1,000,000,000	Dump savepoints dump heap too
SGA	SYSTEMSTATE	2+inc child	System state dump
	HANGANALYZE	1-10	Errorstacks from hung processes
	LIBRARY_CACHE	0..100/ADDR	Dump lib cache/obj at ADDR
	MTSSTATE	15=everything	
	ROW_CACHE	10	Dump the dictionary cache
	LOCKS	10	Locks held by LCK (OPS only)
	ENQUEUES	2+dump resources 3+dump locks	Enqueue information
	LATCHES	2+dump resources 3+dump locks	Latch information
Special Actions	TRACE_BUFFER_ON	size(bytes)	7.3.4+ circular buffer tracing
	ADJUST_SCN	<SCN/1billion>	Dump the current SCN.
	COALESCE	count<<16 | TS	Coalesce Tablespace TS Both 'count' & TS required.
	OPEN_FILES	count<<16 | TS	Force all files to be opened NB: Does not dump any info

표 3 - Event Actions List

Event Name	Event Number	Event Name	Event Number
IMMEDIATE	Do it NOW	SORT_RUN	10033
CONTROL_FILE	10000	SORT_END	10032
DB_FILES	10222	CREATE_REMOTE_RWS	10036
LOGON	10029	ALLOC_REMOTE_RWS	10037
LOGOFF	10030	QUERY_BLOCK_ALLOC	10038
BEGIN	10010	PARSE_SQL_STATEMENT	10035
END	10011	TYPE_CHECK	10039
DEADLOCK	60		

표 4 - Triggering Events

1.5.2. On-Error Dump Events

◆ Dump 를 수행하는 방법은 아래와 같다.

```
# initial Parameter 에 설정
event = "<error> trace name errorstack level <n>"

# alter system 또는 alter session 명령어 사용
alter session set events  '<error>' trace name errorstack level <level>'

# ORADEBUG 사용
oradebug session_event <error> trace name errorstack level <level>
    ➔ 현재 연결된 세션에 관련된 프로세스에 대해 Event를 설정한다. 이 것은 alter
      session 명령어와 같다.
oradebug event <error> trace name errorstack level <level>
    ➔ 이 명령어는 현재 모든 프로세스와 프로세스에 의해 실행된 모든 서브 세션에 대해
      Event를 설정한다.
```

<error> : Server Process 또는 Background Process 에서 발생하는 ORA error 로, 이것에 대한 리포팅 할 수 있다.

level <n> : 각 덤프별로 제공되는 결과의 타입과 양을 결정한다.

- Level 1 : PROCESS STATE 를 추가한다.
- Level 2 : CONTEXT AREA 를 추가한다.

단, ORA-3113, 3114 (end-of-file on communication channel)과 같은 에러는 클라이언트에서 보여주는 에러이므로 on-error Event 를 설정하기에 적절치 않다. 오로지 서버 에러만이 이 이벤트에 영향을 받아 리포팅될 수 있다.

예1) Deadlock 이 발생할 때 errorstack 을 제공(initial Parameter 세팅)

event = "60 trace name errorstack level 1"

예2) ORADEBUG 사용하여 특정프로세스의 PROCESS STATE 덤프를 수행.

(적용할 세션의 PROCESS ID 를 우선 확인한다. 여기서는 3142 임.)

```
SQL> oradebug set 3142
SQL> oradebug unlimit
SQL> oradebug dump errorstack 1
```

1.5.3. Change Behavior Events

◆ 일반적으로 initial parameter 에 설정한다.

◆ 구문 형식은 Trace Events 와 같지만 그 목적은 다르다. 주로 오라클 서버의 특정 기능을 제어하거나, 튜닝을 목적으로 사용된다.

```
event = "<event> trace name context forever, level <level>"
```

<event> : 변경할 기능을 가리킨다.

예) SMON 이 free space 확보하기 위한 단편화 제거(Coalescing)를 막는 Event 설정

event = “10269 trace name context forever, level 10”

1.5.4. Trace Events

◆ 일반적으로 initial parameter 또는 ALTER {SYSTEM | SESSION} 명령어를 사용하여 세팅한다.

```
# initial parameter 에 세팅
event = "<event> trace name context forever, level <level>"

# alter system 또는 alter session 으로 설정
alter {session|system} set events '<event> trace name context forever, level <n>';
```

<event> : 변경할 기능을 가리킨다.

level <n> : 기능을 조정하는데 사용된다.

예1) 10015 Event : rollback segment recovery 정보를 덤프함.(code 10015: undo segment recovery)

event = "10015 trace name context forever,level 10"

예2) 10046 Event : SQL 에 대한 트레이스를 수행하며, sql_trace = true 와 같다.

- Level 1 : 기본적인 SQL 트레이스를 활성화 환다. (DEFAULT)
- Level 4 : Level 1 의 결과에 추가적으로 바인드 변수를 트레이스 한다.
- Level 8 : Level 1 의 결과에 추가적으로 Wait Events 를 트레이스 한다.
- Level 12 : Level 1 의 결과에 바인드 변수 및 Wait Events 를 트레이스 한다.

SQL> alter session set events '10046 trace name context forever, level 12';

2. 이벤트 사용 및 덤프 사례

2.1. Database Crash, Hang and Loop Issues

DBMS 를 운영하다 보면, H/W 또는 OS Kernel 에 관련한 예상치 못한 문제가 발생시 DB 에도 그 영향이 미쳐 Crash 되거나 DB 가 멈추는 문제들이 생길 수 있다. 이런 문제가 발생했을 때 DBA 는 그 원인을 규명하고 분석한 결과를 제시할 수 있어야 한다. 이런 문제가 발생했을 당시 어떤 상태인지 덤프를 수행하고, 그 결과를 분석하는 방법에 대해 알아보도록 하겠다.

2.1.1. 일반적인 Failures

- Crash : Internal errors(ora-600), OS 문제 (세그먼트 결함, BUS 에러(UNIX) , Access violations 등)
- Hang : 프로세스가 아무일도 하지않고 이벤트를 대기하는 상황이다.
- Loop : 프로세스가 특정 오퍼레이션을 무한정 반복하는 경우이다.
- 단순히 시스템이 과부하로 느려지는 경우는 시스템 및 DB 의 튜닝을 고려해야 한다.

2.1.2. 진단 관련 파일들

- alert.log File : 오라클 인스턴에 의해 생성
  - 백그라운드 및 포어그라운드 프로세스의 정보
  - 일반적인 에러에 대한 요약정보와 구체적인 정보가 담긴 트레이스 파일의 위치 정보
- Trace Files : 각 서버프로세스에 대해, 에러 및 예외 상황이 발생할 경우 트레이스 파일에 그것과 관련한 정보를 기록한다.
  - Trace 파일의 헤더에는 기본적인 OS 정보와 DB 정보,그리고 프로세스 정보를 포함한다.
  - Stack Trace : Trace 파일에서 가장 중용한 부분이며, Crash 이전에 문제를 야기한 프로세스에서 수행되었던 Call 정보를 순서대로 표현한다. Stack Trace 는 소스코드와 연관지어 문제를 이해하는데 사용된다.

```
----- Call Stack Trace -----
calling              call     entry              argument values in hex
location             type     point              (? means dubious value)
-------------------- -------- ------------------ ----------------------------
ksedst+001c          bl       ksedst1            000000001 ? 700000010008000 ?
ksm_4031_dump+06b0   bl       ksedst             110562A70 ?
ksmasg+00f0          bl       ksm_4031_dump      FFFFFFFFFFEB580 ? 110195B98 ?
                                                  FFFFFFFFFFEB6E0 ?
                                                  4802208744422587 ?
                                                  100098448 ? 000000000 ?
kghnospc+05e8        bl       _ptrgl
kghalf+0488          bl       kghnospc           000000000 ? 1104D8BE0 ?
                                                  110195B98 ? 000000001 ?
                                                  000000000 ?
.....
```

◆ APPLICATION Log Files : 유저 어플리케이션에서 발생할 수 있는 모든 로그를 말한다. 예를 들어 export/import 시 수행되는 동안 처리된 결과와 에러나 예외사항은 로그 파일에 기록된다. 또한, Unix 상에서 장애시 생성되는 프로세스 메모리 덤프(Core file) 는 debugger 를 사용하여 Stack Trace 를 얻어 분석할 수 있다.

◆ System Log Files : 시스템 로그 파일은 OS 레벨에서의 에러와 예외에 대한 메시지를 캡쳐한다. 주로 하드웨어 또는 OS 관련 문제를 해결하는 데 주로 사용된다. OS 로 인해 DBMS 에도 영향을 줄 수 있지만 DBMS 에서의 잘못된 호출 등으로도 OS error 를 유발할 수도 있다. 시스템 로그 파일의 예로 솔라리스 OS 와 같은 경우 /var/adm/messages 에서 시스템 로그를 확인할 수 있다.

2.1.3. Hang에 대한 진단

◆ Hang 은 프로세스 테이블에 걸쳐 Run Queue 에 들어오는 것이 없이 지속적으로 프로세스 순환이 없을 때 일어난다. 즉, 특정 프로시저는 더 이상 작업을 수행하는 것이 없이 대기하는 상태이다. 보통 CPU 를 거의 사용하지 않는다.

◆ Hang 관련 데이터 수집 및 분석 방법

- process state dump, system state dump, error stack 으로 덤프 수행, 덤프 및 트레이스 파일 분석
- V$SESSION_WAIT, V$LOCK, V$LATCH, V$LATCHHOLDER 등을 통한 WAIT EVENT 분석
  - 특정 세션들이 아무런 응답없이 대기하고 있다면, 어떤 대기 이벤트로 기다리고 있는 지를 확인 할 수 있다.
- HANGANALYZE event 를 사용하여 수집, 분석하는 방법

2.1.4. Looping데 대한 진단

◆ 프로세스는 아무 것도 하지 않는 어떤 것을 기다릴 때 Loop 에 빠질 수 있다.

◆ State Objects 가 자원을 지속적으로 점유하고 있을 때, 이것이 프로세스내 의 어떤 작업 또는 인스턴스 자체의 어떤 오퍼레이션에 의해 발생하고 있는 지 판단해 볼 필요가 있다.

◆ State Objects : SGA 와 연관되어 있는 구성요소들이다. Background 및 Foreground 프로세스, Sessions, Latches 와 Enqueue, buffer handles 이 이에 해당한다. State Objects 는 Process, Session, Transaction 타입으로 분류된다. State Objects 에 문제가 생기는 경우 PMON 이 클린업을 수행한다. 그리고 systemstate 및 processstate 덤프를 수행하여 분석할 수 있다.

◆ Process State Objects : Oracle Processes 와 State Objects 간에는 1 대 1 로 연관지을 수 있다. 단, PSEUDO 프로세스는 제외한다. PSEUDO 프로세스는 Oracle Shared Server 환경에서 세션의 이전을 조절하기 위해 사용된다. 각 플랫폼 별로 프로세스 번호를 에는 차이가 있다.

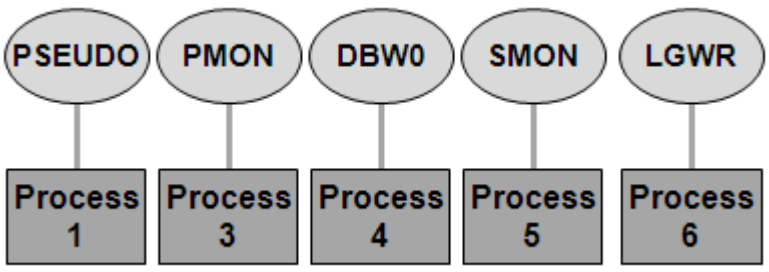

그림 1 - Process State Objects

2.1.5. Process State Dump 수집 및 덤프 구성

1) Processstate dump 수행 방법

```
# Alter session 명령어 사용시
SQL> alter session set events 'immediate
  2  trace name processstate level 10' ;

# 에러에 대한 process state 덤프 수행시 (init.ora 에 설정)
error="<error#> trace name processstate level 10"

# oradebug 사용시
SQL> oradebug setospid <process id>
SQL> oradebug dump processstate 10
```

2) Process State Dump 예제

```
PROCESS STATE
-------------
Process global information:
     process: 70000002f261988, call: 70000002f3ab118, xact: 0, curses: 70000002f36e6c0,
usrses: 70000002f
36e6c0
  ----------------------------------------
  SO: 70000002f261988, type: 2, owner: 0, flag: INIT/-/-/0x00
  (process) Oracle pid=13, calls cur/top: 70000002f3ab118/70000002f3ab118, flag: (0) -
            int error: 0, call error: 0, sess error: 0, txn error 0
  (post info) last post received: 0 0 0
              last post received-location: No post
              last process to post me: none
              last post sent: 0 0 0
              last post sent-location: No post
              last process posted by me: none
    (latch info) wait_event=0 bits=0
    Process Group: DEFAULT, pseudo proc: 70000002f2a6650
    O/S info: user: ora10, term: pts/1, ospid: 827452
    OSD pid info: Unix process pid: 827452, image: oracle@IBM (TNS V1-V3)


--------------------------------------------------------------------------------
SO : State object address
type : 오브젝트 타입에 대한 인덱스 엔트리 (process, session, post info, latch info 등)
owner : 이 state object를 소유하고 있는 state object address
flag : INIT - 오브젝트가 초기화되었음
       FLST - freelist 상에 있는 state object
```

```
      CLN  - PMON에 의해 오브젝트가 해제됨.
DID : Resoure ID
```

3) Processstate/errorstack dump 는 다음과 같은 섹션들로 구성된다.

A. Trace file header : 트레이스 파일 헤더 System, OS, DB 기본 정보를 포함.

B. Current errorstack : 에러와 관련한 스텍 트레이스

C. Current SQL statement : 현 덤프와 연관된 SQL 문

D. Current Call stack : Call 과 관련한 스텍 트레이스

E. Processstate : Process state 덤프 내용

F. Context Area : 문맥에 대한 설명 등.

G. Etc.

```
# Trace file header 의 예

Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /home/ora10/product/db
System name:    AIX
Node name:      IBM
Release:        3
/home/ora10/admin/TEST/udump/ora_ora_827452.trc
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /home/ora10/product/db
/home/ora10/admin/TEST/udump/ora_ora_827452.trc
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /home/ora10/product/db
System name:    AIX
Node name:      IBM
Release:        3
Version:        5
Machine:        000B5CBF4C00
```

```
# Current errorstack 의 예

ksedmp: internal or fatal error
ORA-00600: internal error code, arguments: [2663], [0], [464592], [0], [464858], [], [], []
```

```
# Current SQL statements의 예
```

```
Current SQL statement for this session:
alter database open resetlogs
```

 # Current call stack의 예

```
----- Call Stack Trace -----
calling              call     entry              argument values in hex
location             type     point              (? means dubious value)
-------------------- -------- ------------------ ----------------------------
ksedst+001c          bl       ksedst1            088444844 ? 041144844 ?
ksedmp+0290          bl       ksedst             104A2E270 ?
ksfdmp+0018          bl       03F27F64
kgeriv+0108          bl       _ptrgl
kgesiv+0080          bl       kgeriv             7000000101FDCB0 ? 000000000 ?
                                                 000000020 ? 000000002 ?
                                                 000000001 ?
ksesic4+0060         bl       kgesiv             700000018C8B460 ?
                                                 700000010017A88 ? 000000000 ?
                                                 000000000 ? 000000001 ?
kcrfw_redo_gen+0768  bl       ksesic4            A6700000A67 ? 000000000 ?
                                                 000000000 ? 000000000 ?
                                                 0000716D0 ? 000000000 ?
                                                 000000000 ? 000000000 ?
kcbchg1_main+25d4    bl       kcrfw_redo_gen     1003320C0 ? 000000000 ?
                                                 FFFFFFFFFFEC160 ? 110000E60 ?
                                                 700000010008000 ?
                                                 FFFFFFFFFFEB390 ? 000000010 ?
                                                 70000001A25E268 ?
kcbchg+01b8          bl       kcbchg1_main       FFF00FFFFFEB890 ? 000000430 ?
                                                 FFFFFFFFFFEB8A0 ? 000000430 ?
                                                 FFFFFFFFFFEB8F0 ? 000000430 ?
.....
```

 # Process state 의 예

```
PROCESS STATE
-------------
Process global information:
     process: 70000001a261988, call: 70000001a3aa600, xact: 700000018c341e0, curses:
70000001a381270, usr
ses: 70000001a3827d8
  ----------------------------------------
```

```
  SO: 70000001a261988, type: 2, owner: 0, flag: INIT/-/-/0x00
  (process) Oracle pid=13, calls cur/top: 70000001a3aa600/70000001a3a84b8, flag: (0) -
            int error: 0, call error: 0, sess error: 0, txn error 0
  (post info) last post received: 585 0 4
              last post received-location: kslpsr
              last process to post me: 70000001a25f228 1 22
              last post sent: 0 0 24
              last post sent-location: ksasnd
              last process posted by me: 70000001a25e268 1 6
    (latch info) wait_event=0 bits=0
    Process Group: DEFAULT, pseudo proc: 70000001a2a6650
    O/S info: user: ora10, term: pts/0, ospid: 897036
    OSD pid info: Unix process pid: 897036, image: oracle@IBM (TNS V1-V3)
.....
```

```
# Session Cursor Dump 예


PROCESS STATE
-------------
Process global information:
     process: 70000001a261988, call: 70000001a3aa600, xact: 700000018c341e0, curses:
70000001a381270, usr
ses: 70000001a3827d8
  ----------------------------------------
  SO: 70000001a261988, type: 2, owner: 0, flag: INIT/-/-/0x00
  (process) Oracle pid=13, calls cur/top: 70000001a3aa600/70000001a3a84b8, flag: (0) -
            int error: 0, call error: 0, sess error: 0, txn error 0
  (post info) last post received: 585 0 4
              last post received-location: kslpsr
              last process to post me: 70000001a25f228 1 22
              last post sent: 0 0 24
              last post sent-location: ksasnd
              last process posted by me: 70000001a25e268 1 6
    (latch info) wait_event=0 bits=0
    Process Group: DEFAULT, pseudo proc: 70000001a2a6650
    O/S info: user: ora10, term: pts/0, ospid: 897036
    OSD pid info: Unix process pid: 897036, image: oracle@IBM (TNS V1-V3)
.....
```

2.1.6. System State Dump 수집 및 덤프 구성

1) System State Dump

- System state dump 는 인스턴스의 모든 프로세스들에 대한 Process state dump 이다.
- 문제를 일으키는프로세를 알지 못할 때 유용하다.
- 단, 전체 프로세스에 대해 수행하기 때문에 그 용량이 대단히 클 수 있으므로, MAX_DUMP_FILESIZE 파라메터를 충분히 크게 설정할 필요가 있다. (10g Default : UNLIMITED)
- DB 가 Failures 이 일어났을 때 분석하기 위한 덤프로 활용된다.
- 분석은 Oracle Support 또는 Service Request 로 의뢰할 수 있다. 내부 분석 툴(SSA) 등을 사용함.

2) System State Dump 를 즉시 수행하는 방법

```
SQL> alter session set events 'immediate trace name systemstate level 10';
```

3) 에러와 관련된 System state Dump 를 설정하는 방법(init.ora 에 설정)

```
event = "<error#> trace name systemstate level 10"
```

4) System State Dump 구성

- 일반적인 트레이스 파일 헤더
- 시스템 전체 정보
- 프로세스 정보 – 가장 중요하고, 우선적으로 분석해야 될 부분임. 이 부분에서 첫번째로 "System State" 에 대한 정보가 나오고, 그 다음으로 오라클 백그라운드 프로세스의 Process state objects 에 대한 덤프가 기술된다. 그리고 나서 User Process 의 state objects(session, call, enqueue 등)에 대한 덤프가 기술된다.

```
PROCESS 1:
  ----------------------------------------
  SO: 70000002f25bb08, type: 2, owner: 0, flag: INIT/-/-/0x00
  (process) Oracle pid=1, calls cur/top: 0/0, flag: (20) PSEUDO
            int error: 0, call error: 0, sess error: 0, txn error 0
  (post info) last post received: 0 0 0
              last post received-location: No post
              last process to post me: none
              last post sent: 0 0 0
              last post sent-location: No post
              last process posted by me: none
    (latch info) wait_event=0 bits=0
    O/S info: user: , term: , ospid:  (DEAD)
    OSD pid info: Unix process pid: 0, image: PSEUDO
```

```
Dump of memory from 0x070000002F248480 to 0x070000002F248688
70000002F248480 00000000 00000000 00000000 00000000  [................]
        Repeat 31 times
70000002F248680 00000000 00000000                     [........]
PROCESS 2:
  ----------------------------------------
  SO: 70000002f25c2e8, type: 2, owner: 0, flag: INIT/-/-/0x00
  (process) Oracle pid=2, calls cur/top: 70000002f3a8778/70000002f3a8778, flag: (e) SYSTEM
            int error: 0, call error: 0, sess error: 0, txn error 0
  (post info) last post received: 0 0 33
              last post received-location: ksrpublish
              last process to post me: 70000002f261988 124 0
              last post sent: 0 0 34
              last post sent-location: ksrmdone
              last process posted by me: 70000002f261988 124 0
    (latch info) wait_event=0 bits=0
    Process Group: DEFAULT, pseudo proc: 70000002f2a6650
    O/S info: user: ora10, term: UNKNOWN, ospid: 1122340
    OSD pid info: Unix process pid: 1122340, image: oracle@IBM (PMON)
```

5) 오라클에 접속할 수 없을 때 System State Dump 수행 방법

만약 DB 가 Hang 상태인 경우, 오라클의 지원을 받기 위해 필요한 System state dump 를 취할 필요가 있다. 그런데 이러한 상황에서 오라클에 접속이 불가능한 상태로 빠지는 경우가 있다. 보통은 서버측에서 SYSDBA 로 접속하여 system state dump 를 적어도 3 회이상 수행하도록 요청하고 있지만, SYSDBA 로의 접속마저 어려운 상태라면 다음과 같은 방법을 사용할 수 있다.

```
dbx -a PID (where PID = any oracle shadow process)
dbx() print ksudss(10)
...return value printed here
dbx() detach
```

```
# 우선 shadow process를 찾을 필요가 있다.
$ ps -ef | grep ora
   ora10  229394       1   1   Jun 03      -  2:27 ora_cjq0_ora
   ora10  520416       1   0   Jun 03      -  0:01 ora_arc0_ora
   ora10  524502       1   0   Jun 03      -  0:03 ora_qmnc_ora
   ora10  761862       1   0   Jun 03      -  0:00 ora_reco_ora
   ora10  827554 1044532   1 01:19:02  pts/1  0:00 sqlplus
   ora10  933926       1   0   Jun 03      -  1:44 ora_ckpt_ora
   ora10  979038  827554   1 01:19:03  pts/1  0:00 /usr/bin/ksh
   ora10 1011948       1   0   Jun 03      -  0:40 ora_smon_ora
   ora10 1044532  806932   0 23:04:07  pts/1  0:00 -ksh
   ora10 1052696       1   0   Jun 03      -  0:17 ora_lgwr_ora
   ora10 1061038       1   0   Jun 03      -  0:26 ora_dbw0_ora
   ora10 1089784  979038   0 01:19:06  pts/1  0:00 grep ora
   ora10 1106018  979038   3 01:19:06  pts/1  0:00 ps -ef
```

```
   ora10 1110062       1   0   Jun 03      -  0:11 ora_mman_ora
   ora10 1122340       1   0   Jun 03      -  1:47 ora_pmon_ora
   ora10 1150982       1   0   Jun 03      -  0:02 ora_q001_ora
   ora10 1163440       1   0   Jun 03      -  0:00 ora_q000_ora
   ora10 1179852       1   0   Jun 03      -  0:10 ora_arc1_ora
   ora10 1183788  827554   1 01:19:02      -  0:00 oracleora
(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
   ora10 1188084       1   0   Jun 03      -  8:35 ora_mmnl_ora
   ora10 1204456       1   0   Jun 03      -  0:11 ora_psp0_ora
   ora10 1212486       1   0   Jun 03      -  1:36 ora_mmon_ora

$ dbx -a 1183788
Waiting to attach to process 1183788 ...
Successfully attached to oracle.
warning: Directory containing oracle could not be determined.
Apply 'use' command to initialize source path.

Type 'help' for help.
reading symbolic information ...
stopped in read at 0x90000000005c718 ($t1)
0x90000000005c718 (read+0x1c4) e8410028          ld   r2,0x28(r1)
(dbx) print ksudss(10)
2
(dbx) detach

$ ls -lrt *1183788*
-rw-r-----   1 ora10    dba         2903994 Jun  8 01:20 ora_ora_1183788.trc

$ vi ora_ora_1183788.trc
/home/ora10/admin/TEST/udump/ora_ora_1183788.trc
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /home/ora10/product/db
System name:    AIX
Node name:      IBM
Release:        3
Version:        5
Machine:        000B5CBF4C00
Instance name: ora
Redo thread mounted by this instance: 1
Oracle process number: 13
Unix process pid: 1183788, image: oracle@IBM (TNS V1-V3)

*** 2009-06-08 01:20:36.951
*** ACTION NAME:() 2009-06-08 01:20:36.942
*** MODULE NAME:(sqlplus@IBM (TNS V1-V3)) 2009-06-08 01:20:36.942
*** SERVICE NAME:(SYS$USERS) 2009-06-08 01:20:36.942
*** SESSION ID:(159.905) 2009-06-08 01:20:36.942
==============================================
SYSTEM STATE
------------
System global information:
     processes: base 70000002f247918, size 150, cleanup 70000002f25c2e8
     allocation: free sessions 70000002f2aec50, free calls 0
     control alloc errors: 0 (process), 0 (session), 0 (call)
     PMON latch cleanup depth: 0
     seconds since PMON's last scan for dead processes: 20
     system statistics:
.....
```

2.2. Heap Corruption

보통 RDBMS 와 일반적인 소프트웨어에서는 예정된 작업을 수행하여 어떤 반복적인 결과를 얻어낸다. 그런데 Shared Memory 에서 예상치못한 비정상적인 코드 또는 키조합이 프로세스로 리턴된다면 Heap corruption 을 일으키게 된다. Heap corruption 은 일반적으로 다음을 포함하는 불특정한 문제를 가지고 발생한다.

- ORA-600 : 일반적인 Heap corruption 상황이며, 일관성 체크 실패시 ORA-600 에러를 연관된 arguments 와 trace 파일을 가지고 발생한다. ORA-600 는 H/W 장애, Memory, 버그, disk corruption 등 다양한 문제에 대해 발생한다.
- Core Dump (UNIX) (ORA-7445) / GPFs(NT) : 이 에러는 프로세스가 메모리영역 밖의 특정 주소를 엑세스하려고 시도할 때 주로 발생한다. 대다수가 메모리 포인터의 문제이다.

2.2.1. Heap Corruption 에 대한 정보 수집

- Corrupter 를 식별하기 위해 corruption 위치에 강제로 에러를 발생시키는 방법을 사용할 수 있다.
- **_db_block_cache_protect = {false | true}** : 이 히든 파라메터를 True로 설정하면, 우발적인 쓰기로부터 Buffer Cache 를 보호한다. 설정 후 startup 할 때 각 Buffer 는 read-only 상태로 설정된다. 버퍼를 업데이트 하기 위해서는 명시적으로 read-write 상태로 설정해야 하며, 그 후에 다시 read only 상태로 돌아온다. 그러므로 Buffer Cache의 Corruption을 찾아내는데 도움을 줄 수 있지만, 운영중에는 사용하길 권장하지 않는다. 또한, 추가적인 메모리와 CPU 오버해드가 발생하므로 주의해야 한다.
- **Event 10501** : Oracle9i 이후 10501 이벤트를 수행하여 주기적으로 heap 을 체크할 수 있다. 매 user call 의 끝지점과 PL/SQL 에서 SQL 을 호출하는 지점의 heap 리스트를 확인할 수 있다. 하나의 세션에 대해서만 활성화할 수 있으며, 활성화 되면 타겟이 되는 Heap 이 매번 호출될 때 마다 체크한다. 이 이벤트는 일반적으로 10235 이벤트이 비용이 많이 소요될 때 사용된다.

```
# init.ora 에 설정시
event = '10501 trace name context forever, level nn'

# alter session  명령어 사용시
SQL> alter session set events '10501 trace name context forever, level nn';



# Level : 1 - Top PGA
          2 - Top SGA
          4 - Top UGA
          8 - Current call
         16 - User Call
         32 - Large Pool
```

- **Event 10235 :** Event Name "Check memory manager internal structures"
  이 이벤트를 활성화 하면 heap corruption 을 일으키는 문제의 최종점을 찾을 수 있다. 그리고

Heap Manager 에서 추가적인 체크 수행하게 되며, 몇몇 플랫폼에서는 Memory Protection 이 설정된다. 이 이벤트는 추가적인 CPU 오버헤드가 발생할 수 있으므로 주의해야 한다. 또한 Level > 2 설정하면, OLTP 시스템에 잠재적인 문제를 야기할 수 있다. 10235 이벤트는 세션 레벨에서 설정할 수 없다.

```
# init.ora 에 설정시
event = '10235 trace name context forever, level n'


# Level : 1 - Heap free 사항을 빠르게 체크
          2 - level 1 + 할당/해제에 관련한 junk를 메모리에 채움.
          3 - level 2 + heap 이 해제 될 때 속한 chunks를 확보함.
          4 - level 3 + 해제 가능한 영속적인 chunks를 만듬(체크됨)
        8+N - (N <= 4) Level N에 대한 것과 매 Operation 마다 주어진 heap을 체크한다.
       16XN - 각 Operation 마다 Top PGA(순환적), Top SGA(비순환적)을 체크
```

◆ **Event 10049 :** Event Name : "Protect library cache memory heaps"

이 이벤트를 활성화 화면 Library Cache Object 를 보호를 설정한다. 단, 메모리와 CPU 오버헤드를 발생시킨다.

```
# init.ora 에 설정시
event = '10049 trace name context forever, level <level>'


# Level : 1 - OS page 경계에 Heap extents가 할당된다. 그러나 Page 보호는 이 Page들에
              적용되지 않는다.
         10 - Level 1과 같지만 추가적으로 OS page는 object가 Share 모드로 pin 될 때
              read only 상태로 마크된다. 이것은 프로세스 Pin 상태에서 블록이 쓰여지려
              할 때 Segmentation fault 를 야기한다.
```

2.2.2. Heap Dumps

◆ **Heap Dumps :** Heap 에 대한 덤프를 수행할 때 사용되는 Event

- **heapdump :** PGA, SGA, UGA 덤프

```
# immediate Dump
SQL> alter session set events 'immediate trace name heapdump level <level>'
# On Error Dump
SQL> alter session set events '<error#> trace name heapdump level <level>'

Heap          Level         Level with Contents
Top PGA           1         1025 (1024+1)
Top SGA           2         2050 (2048+2)
Top UGA           4         4100 (…)
Current call      8         8200
User call        16         16400
Large pool       32         32800
```

- **heapdump_addr :** subheap 덤프. UGA dump 상에 ds={address}값

```
SQL> alter session set events 'immediate trace name heapdump_addr <addr>'
```

- **row_cahce :** dictionary cache 덤프

```
SQL> alter session set events 'immediate trace name row_cache level <n>'
```

- **buffers :** buffer cache 덤프

```
#immediate dump
SQL> alter session set events 'immediate trace name buffers level <n>'

# on error dump (init.ora)
event = "<error#> trace name buffers level <n>"

Buffers                <level>      <level>      <l evel>
Description            Headers      Brief Block  Full Block
Block                        1              2           3
and latch lru                4              5           6
and users/waiters            8              9          10
```

- **library_cache :** library cache 덤프 – Level 10 을 사용하면 전체 library cahce 에 대한 덤프를 수행한다.

```
SQL> alter session set events 'immediate trace name library_cache level <addr>'
```

2.3. Data Block 관련 사례(ORA-600 관련 사례)

2.3.1. ORA-600[3339]

Block Corruption 의 일종으로 H/W 문제 또는 Memory corruption 으로 인한 블록의 손상등의 원인으로 발생할 수 있다. 일반적인 데이터 블록이 손상되었을 경우 아래의 이벤트를 적용하여 잘못된 블록이 쓰여지는 것을 막을 수 있다. 만약에 Undo(rollback) Block 이 손상된 경우에는 아래 히든파라메터를 사용하여 인스턴스 recovery 를 막고 불완전한 상태로 강제 오픈할 수 있다. 단 복구 이후 손상된 테이블 및 인덱스의 재구성 또는, Undo 테이블스페이스 재구성이 필요하다.

```
event = "10210 trace name context forever, level 10"  ← Validate Data Block
event = "10211 trace name context forever, level 10"  ← Validate Index Block
_db_block_cache_protect = true
```

Undo(rollback) 블럭이 손상된 경우

```
_offline_rollback_segments = (r01, r02)
_corrupted_rollback_segments = (r01, r02) ← rollback segment 재생성 필요
_allow_resetlogs_corruption = true      ← controlfile 재생성 또는 resetlogs로 강제 오픈할 때
```

2.3.2. ORA-600[ktssdrp1] , [ktsiseginfo1]

DB 운영중에 All redo log 파일 삭제한 경우 발생하며, Dictionary view 의 undo$ & seg$ 의 undo 관련 값이 일치 하지 않을 경우 발생함.

- dba_rollback_segs view 에는 없으나 alert 상에서는 계속해서 Needs Recovery 상태임.
- 관련 Error : ORA-600 [12700], ORA-600 [4097]
- 원인 : Undo Segment 중 문제가 되는 _SYSSMU3$ 가 SEG$ Entry 에 없기 때문
- Undo$ 와 seg$ 에서 일치하지 않는 Undo Segment 찾기

```
select us#, name, ts#, file#, block#,status$
        from undo$
      where (ts#, file#, block#) not in
            (select ts#, file#, block# from seg$ where type# in (1,10)) and status$ > 1;
```

- 해결방법

```
1) DB Cold Backup 할 것
2) initSID.ora 수정
   undo_management=manual
   job_queue_processes =0
   aq_tm_processes=0
   _system_trig_enabled=false
3) DB Shutdown
4) Startup restrict
5) set transaction use rollback segment system;
   만약 set transaction 이 실패할 경우, 더 이상 진행하지 말아야 함.
6) Corrupted 된 undo segment 를 Undo$ 의 status$ 값을 1 (undo segment is droppted)
   값으로 Update 한다.

update undo$ set status$ = 1
where us# = <'undo segment id'>
and ts#=<'tablespace id'>
```

and file#=<'file id'>

and block# = <'block id'>

and status$=2;

-- or --

update undo$ set status$ = 1

where name = <'corrupt undo segment name'>

and status$=2;

This will update only 1 row. Rollback the update if more than 1 row is updated.

7) Commit;

8) Shutdown abort

9) init.ora parameters 를 Step 2 로 원복한다.

Startup restrict

10) 아래 명령으로 Corrupted 된 Tablespace 를 drop:

SQL> drop tablespace <'corrupt undo tablespace name'> including contents and datafiles;

11) Create new undo tablespace

In initSID.ora, set undo_management=auto,

undo_tablespace=<'new undo tablespace name'>

Restart the instance.

12) Take a Cold / Hot backup

2.3.3. ORA-600[ktfdfinsert1]

- Temp Tablespace 가 계속 증가하여 File system Full 난 경우임
- Temp Segments 의 Extent 병합등으로 SMON 에 부하를 줄 수 있음.
- Temp Segments 정렬을 위해 사용될 뿐이므로 Temp Tablespace 를 재구성할 수 있으나 Temp Segments 정리 전까지는 DROP 불가능함.
- Extents 의 병합 및 해제를 막기 위해 이벤트 및 딕셔너리로부터 관련 정보 제거한 뒤 Temp Tablespace 재구성할 수 있음.
- 해결방법

1) Setting Event in init.ora and startup

event="10061 trace name context forever, level 10"

2) Backup the fet$ table

SQL> create table temp_fet as select * from fet$;

3) 해당 1 row delete

SQL> Delete from fet$ where ts#=15 and file#=38 and block#=1172;

4) commit;

SQL> shutdown abort

5) Startup

6) Drop tablespace TEMP2 including contents;

7) Get a clean shutdown with a shutdown immediate and a restart

8) Remove the Event 10061 from the init.ora and then restart the database one more time

9) Take a Backup;

3. 결언

오라클 EVENT 는 오라클이 내부에서 일어나는 모든 사건들을 지칭한다. 실 업무상에서 DBA 가 오라클 DBMS 를 운영하다 보면 여러가지 문제에 직면할 수 있다. 예를 들어, 하나의 세션이 무엇을 위해 대기를 하고 있다든지, DB 내에서 어떤 예상치 못한 현상이 발생했을 때 어떤 과정을 거쳐 발생을 했었는 지 알아 볼 필요가 있을 것이다. 오라클 대기이벤트와 진단이벤트를 사용하면 오라클 내부에서 어떤 일이 일어났었는지 알 수 있다. 또한, DBMS 상에 장애가 발생했을 시, 일반적인 방법으로 운영이 불가능한 경우 EVENT 를 통해 DB 에 영향을 주는 행위를 일으킬 수 있다.

여기서는 문제를 직접적으로 해결하거나 개선할 구체적인 방법을 제시하지는 않았다. 다만, 문제가 발생했을 때 Event 를 사용하여 현 상황에 대한 DUMP 를 수행하고, 수집된 덤프 파일에서 어떻게 구성되어 있고, 문제를 해결하기 위해 필요한 것이 무엇인가를 제시하였다. 수집된 덤프와 트레이스 파일은 오라클 내부로 보내져서 오라클사로부터 분석을 의뢰해야 한다. 그것은 그 자체가 이해하기 어려운 문자들로 구성되어 있어 해석하기 어려우며, 그것을 분석하기 위해 오픈되지 않은 내부 자료 및 툴을 일반 유저 및 고객이 사용할 수 없는 한계가 있기 때문이다.

그렇다고 해서 DBA 가 이런 문제를 고민하지 않을 수 없다. 예측 불가능한 장애 상황은 발생할 것이고, 그것을 해결할 방안을 갖고 있는 경험 많은 DBA 가 되어야 한다. 그리고 오라클의 도움을 받기 위해서도 문제를 해결하기 위한 충분히 필요한 정보를 제공해야만 할 것이다. 그렇게 한다면 오라클 RDBMS 를 좀 더 깊게 이해하고, 가장 현 시스템에 최적화된 DB 를 세팅하여 운영 할 수 있을 것이라 생각한다.