# Goodus 기술노트 [22 회]
# AUDIT

| | |
|---|---|
| **Author** | **고형덕, 노재구** |
| **Creation Date** | **2007-08-01** |
| **Last Updated** | **2007-08-06** |
| **Version** | **1.1** |
|  | |

| Version | 변경일자 | 변경자(작성자) | 주요내용 |
|---|---|---|---|
| 1 | 2007-08-06 | 노재구 | 문서 최초 작성 |
| 2 | | | |
| 3 | | | |

Goodus
Breathing Life into Technology

# Contents

# Audit

## 1.1. Audit 란?

Auditing 기능은 사용자의 행동을 감시하거나 데이터베이스에 관한 통계자료를 얻는 목적으로 사용된다. Auditing 기능을 사용함으로써 누가 어떠한 테이블을 언제 사용하고, 언제 어떤 작업을 하는지를 기록할 수 있다.
데이터베이스 사용자는 일정한 권한을 부여 받아 데이터를 조작(Insert, Update, Delete)하거나 조회(Select) 할 수 있다. 권한을 받은 사용자는 주어진 권한을 이용하여 원래의 목적에 맞지 않는, 접근해서는 안되는 중요한 데이터를 조회하거나 변경할 수도 있다. 이러한 일들을 막기 위해서 사용자가 데이터를 조회, 조작할 때마다 이에 대한 정보를 기록하여, 누가 언제, 무엇을 했는지 확인하는 방법이 필요하다. 이것을 '감사(Audit)'라고 한다.

## 1.2. Database Audit Overview

데이터베이스에 영향을 끼치는 작업을 감시하거나 특정 데이터베이스 작업에 대한 데이터를 모니터하고 수집한다. 이벤트에 대한 정보는 Audit Trail 에 저장된다.

- Database Aduting 을 통해 데이터베이스의 모든 연결에 대해 감사가 가능하다. 승인되지 않은 사용자가 테이블에서 데이터를 삭제하고 있는 경우, DBA 는 데이터베이스에 있는 테이블에서의 행에 대한 성공적인 삭제 여부를 감시하는데 사용된다.
- 특정 데이터베이스 작업을 모니터링하고, 그에 대한 데이터를 모으는 데 사용될 수 있다. 예를 들어, DBA 는 갱신중인 테이블, 수행한 논리적 I/O 횟수 및 시스템이 바쁜 시간에 연결한 동시 User 수에 대한 통계를 수집할 수 있다.
- DBA 에 의해 활성화 또는 비활성화할 수 있다.
- 컬럼의 값에 대한 기록은 불가능하다.

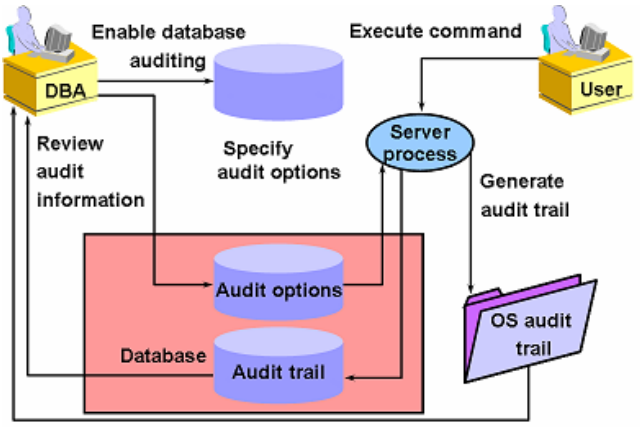



그림 1 Database Audit

1. Enable/Disable Database Auditing (Setting Audit Parameter)
   - 데이터베이스 감사(Database Auditing) 설정은 DBA 에 의해 가능하다. Init 파라메터 중 AUDIT_TRAIL 을 이용하여 데이터베이스 감사 기능을 활성화 또는 비활성화 시킬 수 있다.

| AUDIT_TRAIL Value | Description |
|---|---|
| TRUE / DB | Audit 를 활성화함. Audit 결과는 SYS.AUD$에 저장 |
| DB_EXTENDED | Audit Trail 정보에 SQLBIND, SQLTEXT 추가됨.<br>(Oracle Database 10g New Feature) |
| OS | OS 에서 허용하는 경우 활성화, Audit 결과는 파일로 저장됨. |
| FALSE / NONE | Audit 를 비활성화함. |

표 1 - AUDIT_TRAIL Parameter Values

## 2. Specify Audit Options

- Audit 명령을 사용하여 Audit 할 명령, User, Object, Privilege 를 지정한다. Audit Record 가 발생할 때마다 생성할 지, Session 당 한번 생성할 지를 결정할 수 있다.
즉, 다음과 같은 3 가지 종류가 있다.

■ **Statement Auditing** – SQL 명령문의 유형에 따른 Auditing 설정

ex) AUDIT TABLE BY SCOTT BY ACCESS WHENEVER SUCCESSFUL;
: scott 유저가 테이블에 관련된 명령(create table, drop table 등)이 성공한 경우 기록된다.

■ **Privilege Auditing** – 사용되는 PRIVILEGE 에 따른 Auditing 설정

ex) AUDIT CREATE TABLE BY SCOTT BY SESSION;
: scott 유저가 'create table' 권한이 필요한 명령을 수행시 기록

■ **Object Auditing** – 특정 스키마의 개체의 명령문에 대한 Auditing 설정

ex) AUDIT ALL ON SCOTT.EMP;
: scott.emp 테이블에 대한 모든 명령(select, insert, update, delete, drop 등)에 관한 사항이 기록된다.

설정된 AUDIT 기능은 NOAUDIT 명령으로 제거할 수 있다.
ex) NOAUDIT ALL ON SCOTT.EMP;

## 3. Execute Command

- 사용자가 SQL 또는 PL/SQL 문을 실행할 때 Server Process 는 Audit 옵션을 검색하여 실행중인 Statement 가 감사 대상에 포함되는지 여부를 먼저 결정한다.

## 4. Generate Audit Trail

- audit_trail 파라메터에 정의된 값에 따라 OS 의 파일 또는 데이터베이스 내에 SYS.AUD$에 Audit Trail 레코드를 생성한다. 이 작업은 사용자의 Transaction 과 무관하므로 Transaction Rollback 이 수행될지라도 Audit Trail 레코드는 그대로 유지된다. 단, Audit Trail 레코드는 구문의 execute 단계에서 생성되므로 Parsing 단계에서 오류가 발생하면 생성되지 않는다.

## 5. Review Audit Information

- Audit 를 통해 생성된 정보는 아래의 Audit Trail Data Dictionary 뷰를 통해 확인할 수 있다. OS 에 생성된 Audit Trail 인 경우 OS Utility 를 사용하여 확인한다.
이 정보를 토대로 의심이 가는 작업을 확인하거나 데이터베이스 작업에 대한 모니터링할 수 있다.

| View | Description |
|---|---|
| STMT_AUDIT_OPTION_MAP | Audit 옵션 타입 코드에 대한 정보를 보여준다. |
| AUDIT_ACTIONS | Audit Trail Action 타입 코드에 대한 설명을 포함한다. |
| ALL_DEF_AUDIT_OPTS | Object 가 생성될 대 적용될 수 있는 기본적인 Object-auditing 옵션을 보여준다. |
| DBA_STMT_AUDIT_OPTS | 시스템과 사용자에 걸쳐 현재 시스템의 Audit 옵셥을 보여준다. |
| DBA_PRIV_AUDIT_OPTS | 시스템과 사용자에 걸쳐 Audit 되고 있는 현재 시스템 권한을 보여준다. |
| DBA_OBJ_AUDIT_OPTS<br>USER_OBJ_AUDIT_OPTS | 모든 Object 에 대한 Audit 옵션을 보여준다.<br>USER 뷰는 현재 사용자가 소유한 모든 Object 의 Audit Option 을 보여준다. |
| DBA_AUDIT_TRAIL<br>USER_AUDIT_TRAIL | 모든 Audit Trail 엔트리를 리스트한다.<br>USER 뷰는 현재 사용자에 관련된 Audit Trail 엔트리를 보여준다. |
| DBA_AUDIT_OBJECT<br>USER_AUDIT_OBJECT | 시스템의 모든 Object 에 대한 Audit Trail 레코드를 포함한다.<br>USER 뷰는 현재 사용자가 접근할 수 있는 Object 와 관련된 구문에 대한 Audit Trail 레코드를 리스트한다. |

| | |
|---|---|
| DBA_AUDIT_SESSION<br>USER_AUDIT_SESSION | CONNECT 와 DISCONNECT 에 관련된 모든 Audit Trail 레코드를 리스트한다.<br>USER 뷰는 현재 사용자에 대한 연결 설정과 해제에 관련된 모든 Audit Trail 레코드를 리스트한다. |
| DBA_AUDIT_STATEMENT<br>USER_AUDIT_STATEMENT | 데이터베이스 전반적으로 GRANT, REVOKE, AUDIT, NOADUIT, ALTER SYSTEM 구문과 관계되는 Audit Trail 레코드를 리스트한다.<br>USER 뷰는 현재 사용자와 관련된 내용만을 보여준다. |
| DBA_AUDIT_EXISTS | AUDIT EXISTS 와 AUDIT NOT EXISTS 와 관련된 Audit Trail 엔트리를 리스트한다. |

표 2 - Audit Trail Data Dictionary Views

## 1.3. 예제 1 (AUDIT_TRAIL=DB)

```
SQL*Plus: Release 9.2.0.4.0 - Production on Thu Aug 2 15:25:28 2007

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> show parameter audit_trail
NAME                                 TYPE                 VALUE
------------------------------------ -------------------- ------------------------------
audit_trail                          string               DB

-- Object Audit 설정

SQL> select owner, object_name, object_type, sel from dba_obj_audit_opts
2  where object_name = 'EMP' and owner = 'SCOTT';

OWNER      OBJECT_NAME     OBJECT_TYPE     SEL
---------- --------------- --------------- ----------
SCOTT      EMP             TABLE           -/-

SQL> audit select on scott.emp by session whenever successful;

Audit succeeded.

SQL> select owner, object_name, object_type, sel from dba_obj_audit_opts
2  where object_name = 'EMP' and owner = 'SCOTT';

OWNER      OBJECT_NAME     OBJECT_TYPE     SEL
---------- --------------- --------------- ----------
SCOTT      EMP             TABLE           S/S

SQL> select * from dba_obj_audit_opts
  2  where object_name = 'EMP' and owner = 'SCOTT';

OWNER OBJECT_NAME  OBJECT_TYPE  ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ------------ ------------ --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
SCOTT EMP          TABLE        -/- -/- -/- -/- -/- -/- -/- -/- -/- S/S -/- -/- -/- -/- -/- -/-

* dba_obj_audit_opts 는 object에 설정할 수 있는 각 audit option을 column으로 하고 있으며, 내용은
  [A/S]/[A/S]의 형식을 갖는다. / 의 앞부분은 successful일 경우, 뒷부분은 not successful일 경우를 표시한다.
* A는 by access, S는 by session을 의미한다.
  by access는 해당 명령이 내려질 때마다 정보를 기록하고, by sessoin은 접속된 세션에 대하여 하나의
  레코드만 생성한다.
```

```
SQL> select count(*) from sys.aud$;

  COUNT(*)
----------
         0

SQL> connect scott/tiger
Connected.
SQL> select * from emp where sal > 3000; → 인사 급여테이블에

 EMPNO ENAME      JOB              MGR HIREDATE       SAL  COMM DEPTNO
------ ---------- ------------- ------ --------- ------- ----- ------
  7839 KING       PRESIDENT            17-NOV-81    5000            10

SQL> connect / as sysdba
Connected.

SQL> select count(*) from sys.aud$;

  COUNT(*)
----------
         1

SQL> select os_username, username, timestamp, owner, obj_name, action_name, ses_actions, returncode
  2  from dba_audit_object;

OS_USERNAME USERNAME   TIMESTAMP OWNER OBJ_NAME   ACTION_NAME          SES_ACTIONS          RETURNCODE
----------- ---------- --------- ----- ---------- -------------------- -------------------- ----------
oracle      SCOTT      02-AUG-07 SCOTT EMP        SESSION REC          ---------S------             0

* returncode 가 0 인 경우는 SUCCESS 임을 의미한다.
* by session 로 설정한 경우 audit 를 설정한 경우 action_name 은 'SESSION REC' 으로 표시된다.
* ses_actions columns 은 총 13 가지의 action(alter, audit, comment, delete, grant, index, insert, lock,
  rename, select, update, reference, execute)에 대한 Audit 정보를 포함한다. 각 Action 에 대해 모두
  성공하였다면 'S' 를 실패하였다면 'F' 로 표현되며 성공과 실패가 모두 있었다면 'B' 로 표현된다.
* by access 로 설정한 경우 action_name 컬럼에 해당 action(예: delete, insert)등이 바로 기술된다.


-- Private Audit 설정
   SQL> select * from dba_priv_audit_opts;

   USER_NAME  PROXY_NAME PRIVILEGE            SUCCESS              FAILURE
   ---------- ---------- -------------------- -------------------- ----------
                         CREATE ANY RULE      BY ACCESS            NOT SET
                         ALTER ANY RULE       BY ACCESS            NOT SET
                         EXECUTE ANY RULE     BY ACCESS            NOT SET

   SQL> audit create session by scott;

   Audit succeeded.

   SQL> select * from dba_priv_audit_opts;

   USER_NAME  PROXY_NAME PRIVILEGE            SUCCESS              FAILURE
   ---------- ---------- -------------------- -------------------- ----------
   SCOTT                 CREATE SESSION       BY ACCESS            BY ACCESS
                         CREATE ANY RULE      BY ACCESS            NOT SET
                         ALTER ANY RULE       BY ACCESS            NOT SET
```

```
                    EXECUTE ANY RULE     BY ACCESS           NOT SET
-- connect session
SQL> connect scott/tiger
Connected.

-- disconnect session
SQL> disconnect

Disconnected from Oracle9i Enterprise Edition Release 9.2.0.4.0 - 64bit Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> ! date
Thu Aug  2 18:09:24 KORDT 2007

SQL> connect /as sysdba
Connected.

SQL> alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> select os_username, username, timestamp, action_name, logoff_time, logoff_lread, logoff_pread
  2  from dba_audit_session
  3  where username = 'SCOTT';

OS_USERNAME USERNAME   TIMESTAMP           ACTION_NAME LOGOFF_TIME         LOGOFF_LREAD LOGOFF_PREAD
----------- ---------- ------------------- ----------- ------------------- ------------ ------------
oracle      SCOTT      2007-08-02 18:08:56 LOGOFF      2007-08-02 18:08:59           76            0

-- 다른 쪽 터미널에서 Scott 유저로 접속하고 다시 조회하면,

SQL> select os_username, username, timestamp, action_name, logoff_time, logoff_lread, logoff_pread
  2  from dba_audit_session
  3  where username = 'SCOTT';

OS_USERNAME USERNAME   TIMESTAMP           ACTION_NAME LOGOFF_TIME         LOGOFF_LREAD LOGOFF_PREAD
----------- ---------- ------------------- ----------- ------------------- ------------ ------------
oracle      SCOTT      2007-08-02 18:08:56 LOGOFF      2007-08-02 18:08:59           76            0
oracle      SCOTT      2007-08-02 18:55:59 LOGON
-- Statement Aduit 설정
-- 존재하지 않는 오브젝트에 대한 구문을 실행시에 Audit Trail 작성
SQL> audit not exists by test;

SQL> select * from dba_stmt_audit_opts ;

USER_NAME       PROXY_NAME AUDIT_OPTION         SUCCESS    FAILURE
--------------- ---------- -------------------- ---------- ----------
SCOTT                      NOT EXISTS           BY ACCESS  BY ACCESS

SQL> conn scott/tiger
Connected.

SQL> grant select on not_exist_table to tester;
grant select on not_exist_table to tester
                *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> select os_username,username,timestamp,obj_name,action_name,
  2  obj_privilege, grantee
```

```
  3  from dba_audit_statement;

OS_USERNAME USERNAME   TIMESTAMP OBJ_NAME        ACTION_NAME     OBJ_PRIVILEGE       GRANTEE
----------- ---------- --------- --------------- --------------- ------------------- ----------
oracle      SCOTT      04-AUG-07 NOT_EXIST_TABLE GRANT OBJECT    ---------Y------    TESTER
```

* 3 가지 타입의 Audit Trail 이 생성되는 예제를 보았듯이 누가 언제 무슨 작업을 하겨고 했는지 알 수 있다. 하지만, 위의 결과에서는 어떤 구문을 실행했을 때 Auditing 되었는지 정확히 알 수가 없고, Object Audit 에서 볼 수 있듯이 어떤 데이터를 조회했는지 알 수가 없다.

## 1.4. 예제 2 (AUDIT_TRAIL=DB_EXTENDED, Oracle Database 10g New Feature)

```
SQL> show parameter audit_trail

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
audit_trail                          string      DB_EXTENDED

SQL> connect /as sysdba
Connected.

SQL> audit insert on scott.emp by access;

Audit succeeded.

SQL> select owner, object_type, object_name, ins from dba_obj_audit_opts
  2  where owner = 'SCOTT' and object_name = 'EMP';

OWNER      OBJECT_TYPE     OBJECT_NAME     INS
---------- --------------- --------------- -----
SCOTT      TABLE           EMP             A/A


SQL> desc dba_audit_object

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OS_USERNAME                                        VARCHAR2(255)
 USERNAME                                           VARCHAR2(30)
 USERHOST                                           VARCHAR2(128)
 TERMINAL                                           VARCHAR2(255) client pc 명:사용자를 짐작할 수 있음
 TIMESTAMP                                          DATE
 OWNER                                              VARCHAR2(30)
 OBJ_NAME                                           VARCHAR2(128)
 ACTION_NAME                                        VARCHAR2(28)
 NEW_OWNER                                          VARCHAR2(30)
 NEW_NAME                                           VARCHAR2(128)
 SES_ACTIONS                                        VARCHAR2(19)
 COMMENT_TEXT                                       VARCHAR2(4000)
 SESSIONID                                 NOT NULL NUMBER
 ENTRYID                                   NOT NULL NUMBER
 STATEMENTID                               NOT NULL NUMBER
 RETURNCODE                                NOT NULL NUMBER
 PRIV_USED                                          VARCHAR2(40)
 CLIENT_ID                                          VARCHAR2(64)
```

```
ECONTEXT_ID                                       VARCHAR2(64)
SESSION_CPU                                       NUMBER
EXTENDED_TIMESTAMP                                TIMESTAMP(6) WITH TIME ZONE
PROXY_SESSIONID                                   NUMBER
GLOBAL_UID                                        VARCHAR2(32)
INSTANCE_NUMBER                                   NUMBER
OS_PROCESS                                        VARCHAR2(16)
TRANSACTIONID                                     RAW(8)
SCN                                               NUMBER
SQL_BIND                                          NVARCHAR2(2000)
SQL_TEXT                                          NVARCHAR2(2000)
```

* ORACLE Database 10g 에서 AUD$ 테이블에는 위 DBA 뷰에서 보듯이 SQL_BIND 와 SQL_TEXT 두 컬럼이 추가되었다.

```
SQL> select username, owner, obj_name, action_name, ses_actions, returncode, timestamp,
  2    sql_bind, sql_text
  3  from dba_audit_object;

no rows selected

SQL> conn scott/tiger
Connected.

SQL> variable empno number;
SQL> variable ename varchar2(8);
SQL> begin
  2  :empno := 9999;
  3  :ename := 'TESTER';
  4  end;
  5  /

PL/SQL procedure successfully completed.

SQL> insert into emp values (:empno, :ename, 'MANAGER', 7499, SYSDATE, 3000, '', 20);

1 row created.

SQL> commit;

Commit complete.

SQL> connect / as sysdba
Connected.

SQL> select username, owner, obj_name, action_name, ses_actions, returncode, timestamp,
  2    sql_bind, sql_text
  3  from dba_audit_object;

USERNAME  OWNER  OBJ_NAME  ACTION_NAME SES_ACTIONS RETURNCODE TIMESTAMP    SQL_BIND
--------- ------ --------- ----------- ----------- ---------- ------------ --------------------------
SQL_TEXT
-------------------------------------------------------------------------------------------------------
SCOTT     SCOTT  EMP       INSERT                           0 05-AUG-07    #1(4):9999 #2(6):TESTER
insert into emp values (:empno, :ename, 'MANAGER', 7499, SYSDATE, 3000, '', 20)
```

## 1.5. Controlling the Growth and Size of the Standard Audit

Audit Trail 이 쌓이는 SYS.AUD$ 테이블은 SYSTEM 테이블스페이스 생성된다. 그러므로 SYS.AUD$ 테이블에 데이터가 쌓임으로써 SYSTEM 테이블에 대하여 경합 등 부담을 줄 수 있다. 예를 들어, CREATE SESSION 에 대한 AUDITING 을 수행할 때 세션의 사용량에 따라 엄청난 데이터가 축적될 수 있다. 그러므로 불필요한 AUDIT 의 사용은 데이터베이스의 성능을 저하시킬 수 있다. 그러므로 불필요한 Audit 기능은 제한하고, Audit 하는 기간 동안 SYS.AUD$ 테이블의 크기를 적절히 유지하도록 해야한다.

- Purging Audit Record from the Audit Trail
- Archiving Audit Trail Information
- Reducing the Size of the Audit Trail

제어 방법에 대한 예시는 다음과 같다.

```
-- Purging Audit Record from the Audit Trail

DELETE FROM SYS.AUD$;
DELETE FROM SYS.AUD$ WHERE obj$name =  'EMP' ;
TRUNCATE TABLE SYS.AUD$;

-- Archiving Audit Trail Information

INSERT INTO table SELECT … FROM SYS.AUD$ …
exp W'"sys/sys as sysdba"W' file=aud.dmp tables=aud$;

-- Reducing the Size of the Audit Trail
   : Archiving 과 Purging 을 이용한 관리

  1.  SYS.AUD$ 에 대한 백업을 수행한다. - Export 등
  2.  sys 유저로 접속한다. - 관리자에 의한 수행
  3.  TRUNCATE TABLE SYS.AUD$; (HWM 의 이동으로 extent 할당을 해제하여 테이블 내 공간을 줄임.)
  4.  Audit Trail Record 가 많이 생성되면 1 번부터 재수행한다.
```

## 1.6. Trigger 를 이용한 Audit

앞서 말했듯이 10g 이전의 Audit 의 기능에는 바운드 변수(SQL_BIND)와 구문(SQL_TEXT)를 알 수 있는 방법이 없었다. 그리고 Oracle 9i Database 에서 FGA(Find-Grained Auditing)을 사용하면 좀 더 다양하고 세부적인 Auditing 이 가능하다. 우선, FGA 를 사용하지 않고 구현할 수 있는 방법을 알아 보겠다. 그리고 FGA 는 뒤에서 설명하기로 하겠다.

낮은 버전의 오라클(8i 이하)이나 오라클이 제공하지 않는 AUDITING 을 구현하기 위해서 Trigger 를 이용할 수 있다. 이는 Trigger 를 사용하여 데이터의 액세스나 조작이 일어나기 전에 그 내용을 관리자가 관리하는 테이블로 저장하는 방식이다. 단순한 AUDITING 또는 AUDIT 로 인한 부담을 줄이기 위해 데이터베이스 사용자에 의해 개발된 AUDIT 기능이라고 보면 될 것이다.

트리거를 이용하여 사용자에 의한 트랜잭션에 대한 로그를 남길 수 있다. 하지만 자칫 Trigger 의 내용으로 인해 실제 구문의 수행시간 및 리소스 사용에 영향을 최소한으로 줄여야 한다.

다음은 트리거를 이용하여 특정 테이블의 변경된 내용을 Auditing 하는 예제이다.

1. AUDIT 테이블 생성.

```
connect / as sysdba
Connected.

CREATE TABLE emp_audit (
    old_empno NUMBER(4),
    old_ename VARCHAR2(10),
    old_job VARCHAR2(9),
    old_mgr NUMBER(4),
    old_hiredate DATE,
    old_sal NUMBER(7,2),
    old_comm NUMBER(7,2),
    old_deptno NUMBER(2),
    new_empno NUMBER(4),
    new_ename VARCHAR2(10),
    new_job VARCHAR2(9),
    new_mgr NUMBER(4),
    new_hiredate DATE,
    new_sal NUMBER(7,2),
    new_comm NUMBER(7,2),
    new_deptno NUMBER(2),
    changed_by VARCHAR2(8),
    change_type CHAR(1),
    timestamp DATE );

Table created.
```

2. TRIGGER 생성.

```
CREATE OR REPLACE TRIGGER LogEmpChanges
    BEFORE INSERT OR DELETE OR UPDATE ON scott.emp
FOR EACH ROW
DECLARE
    v_ChangeType CHAR(1);
BEGIN
    /* Use 'I' for an INSERT, 'D' for DELETE, and 'U' for UPDATE. */
    IF INSERTING THEN
        v_ChangeType := 'I';
    ELSIF UPDATING THEN
        v_ChangeType := 'U';
    ELSE
        v_ChangeType := 'D';
    END IF;

    INSERT INTO emp_audit (
        change_type, changed_by, timestamp, old_empno,
        old_ename, old_job, old_mgr, old_hiredate, old_sal,
        old_comm, old_deptno, new_empno, new_ename, new_job,
        new_mgr, new_hiredate, new_sal, new_comm, new_deptno)
    VALUES (
        v_ChangeType, USER, SYSDATE, :old.empno, :old.ename,
        :old.job, :old.mgr, :old.hiredate, :old.sal, :old.comm,
        :old.deptno, :new.empno, :new.ename, :new.job, :new.mgr,
        :new.hiredate, :new.sal, :new.comm, :new.deptno);
END LogEmpChanges;
/

PL/SQL procedure successfully completed.
```

```
connect scott/tiger
INSERT INTO emp VALUES ( 9999, 'TESTER', 'CLERK', 7782, SYSDATE, 1000, 0, 10);
COMMIT;
UPDATE emp SET comm = 300 WHERE empno = 9999;
COMMIT;
DELETE FROM emp WHERE empno = 9999;
COMMIT;

connect /as sysdba

SELECT OLD_EMPNO, OLD_ENAME, OLD_COMM, NEW_EMPNO, NEW_ENAME, NEW_COMM, CHANGED_BY, CHANGE_TYPE,
TIMESTAMP FROM EMP_AUDIT;

OLD_EMPNO OLD_ENAME    OLD_COMM  NEW_EMPNO NEW_ENAME    NEW_COMM CHANGED_BY       CH TIMESTAMP
---------- ---------- ---------- ---------- ---------- ---------- ---------------- -- ---------
                                  9999 TESTER                  0 SCOTT            I  01-AUG-07
      9999 TESTER                0       9999 TESTER          300 SCOTT            U  01-AUG-07
      9999 TESTER              300                               SCOTT            D  01-AUG-07
```

* Before Trigger 로 인하여 사용자가 DML 작업을 수행할 때 해당 테이블의 변경 전후의 column 값 및 변경된 시간 등을 emp_audit 테이블에 기록한다.

* 부록 1.에는 위와 같은 소스를 자동 생성해주는 샘플 Package를 작성해 보았다. 해당 테이블의 정보를 추출하여 audit trail record를 담을 테이블과 이벤트를 처리할 트리거를 자동생성하고 제거할 수 있는 패키지이다. 다시 말해, 편의상 구문을 자동 생성하고, 실행시키는 루틴을 포함시켜 놓았다. 오라클의 제공하는 Audit 기능을 대체할 수 있도록 응용해 보기 바란다.

# 2. FGA(Fine-Grained Auditing)

## 2.1. Fine-Grained Auditing 이란?

데이터베이스 감사(Database Auditing) 기능을 사용하여 특정 스키마 내의 테이블에 SELECT 에 관한 Auditing을 설정하였을 경우 해당 테이블의 조회 관련된 모든 수행기록이 Audit Trail Record의 형태로 저장될 것이다. 이런 경우 실제로 원하지 않는 결과까지도 모두 기록되어 Audit Trail의 양은 상당히 많아 질 것이다. 그리고 이들 데이터에서 특정 컬럼에 대한 조회내역 직접적으로 사용된 SELECT문을 찾기는 어려울 것이다.

FGA(Fine-Grained Auditing)은 이러한 문제에 대한 해결책을 제시했다. Oracle 9i Database 에 등장한 FGA는 특정 데이터를 조회하는 경우에만 감사가 가능하도록 설정할 수 있다. Oracle Database 10g에서는 Oracle 9i Database 에서 SELECT 만 가능했던 FGA의 영역을 DML까지 확장하며 완성되었다.

## 2.2. Fine-Grained Auditing 특징

- 더 상세한 레벨의 감사 기능을 제공한다. 선택적인 감사를 위한 조건으로 SQL의 WHERE절을 기반으로 한다.
- 값에 의한 감사뿐만 아니라 특정 컬럼의 참조 또는 액세스 여부에 대해서도 Auditing이 가능하다.
- DBMS_FGA 패키지를 이용하여 활성화/비활성화 한다.
- CBO(Cost Based Optimizer)인 경우에 정상적으로 작동한다. (인스턴스 레벨의 CBO 설정 및 테이블에대한 분석(Analyze) 이 되어 있어야 하며 SQL에 힌트가 없어야 한다.)
- Oracle Database 9i에서는 SELECT문에서만 가능하며, Oracle Database 10g에서는 Insert, Update, Delete, Merge문에 대해서도 가능하다.
- Oracle Database 10g에서는 컬럼에 대한 옵션(audit_column_opts) 지정으로 DBMS_FGA.ALL_COLUMNS / DBMS_FGA.ANY_COLUMNS를 사용할 수 있다.
- FGA에서 Audit Trail Record는 SYS.FGA_LOG$에 저장된다.

| Parameter Name | Data Types | Description |
|---|---|---|
| object_schema | VARCHAR2 | 설정하고자 하는 스키마명 |
| object_name | VARCHAR2 | 설정하고자 하는 오브젝트명(테이블, 뷰명) |
| policy_name | VARCHAR2 | Policy 명(Unique) |
| audit_condition | VARCHAR2 | 상세 감사 조건, 스키마 오브젝트에 대한 논리적인 데이터 그룹에 상응하는 WHERE절 지칭 SQL문의 조건절에 해당하는 부분이 명시적 묵시적으로 이 조건에 만족할 경우 Audit Trail을 생성함. 조회 결과의 일부가 조건에 해당되는 경우도 Audit Trail을 생성함 |
| audit_column | VARCHAR2 | 감사대상이 되는 컬럼을 지정함. Oracle 9i Database에서는 한 개의 컬럼만 지정이 가능했으나, Oracle Database 10g부터는 하나 이상의 컬럼의 지정이 가능. |
| handler_schema | VARCHAR2 | FGA에 의해 Audit Trail이 생성될 때 실행시킬 프로시저의 소유자 |
| handler_module | VARCHAR2 | FGA에 의해 Audit Trail이 생성될 때 실행시킬 프로시저<br>생성패턴: PROCEDURE <fname>{ obect_schema VARCHAR2, object_name VARCHAR2, policy_name VARCHAR2} AS … |
| enable | BOOLEAN | FGA 활성화여부 TRUE로 지정해야 활성화됨. |
| statement_type(10g) | VARCHAR2 | FGA가 작동하는 쿼리 타입지정(INSERT/DELETE/UPDATE, SELELCT) |
| audit_trail(10g) | BINARY_INTEGER IN DEFAULT | 'AUDIT_TRAIL=> DBMS_FGA.DB_EXTENDED' 로 설정되는 경우 SYS.FGA_log$의 LSQLTEXT, LSQLBIND 컬럼에 Audit Trail이 생성된다. |
| audit_column_opts(10g) | BINARY_INTEGER IN DEFAULT | AUDIT_COLUMN에 나열된 컬럼 중에서 나열된 컬럼 중 모두 엑세스 될 때 Audit Trail을 생성하도록 하는 DBMS_FGA.ALL_COLUMNS 옵션과 AUDIT_COLUMN에 나열된 컬럼중에서 하나라도 엑세스될 때 AUDIT TRAIL이 생성되도록 지정하는 DBMS_FGA.ANY_COLUMNS 옵션이 있다. |

표 3 DBMS_FGA.ADD_POLICY 프로시져

| Parameter Name | Data Types | Description |
|---|---|---|
| object_schema | VARCHAR2 | 해제하고자하는 스키마명 |
| object_name | VARCHAR2 | 해제하고자하는 오브젝트명(테이블, 뷰명) |
| policy_name | VARCHAR2 | Policy 명 |

표 4 DBMS_FGA.DROP_POLICY

## 2.3. Oracle 9i Database FGA 예제

```
CONNECT system/manager

Connected.

CREATE USER scott IDENTIFIED BY tiger
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp;

User created.

GRANT CONNECT, RESOURCE TO scott;

Grant succeeded.

CREATE USER alice IDENTIFIED BY alice
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp;

User created.

GRANT CONNECT, RESOURCE TO alice;

Grant succeeded.

CONNECT scott/tiger

Connected.

@?/sqlplus/demo/demobld.sql

SQLPLUS scott/tiger

Connected.

ANALYZE TABLE emp COMPUTE STATISTICS;

Table analyzed.

ANALYZE TABLE dept COMPUTE STATISTICS;

Table analyzed.

GRANT SELECT ON emp TO alice;

Grant succeeded.

GRANT SELECT ON dept TO alice;

Grant succeeded.
```

```
CONNECT system/manager
Connected.

-- FGA 설정
EXECUTE DBMS_FGA.ADD_POLICY( object_schema => 'SCOTT', -
                             object_name => 'emp', -
                             policy_name => 'aud_emp_sal_mgr', -
                             audit_condition => 'deptno in (10, 20) ', -
                             audit_column => 'sal, mgr', -
                             handler_schema => 'system', -
                             handler_module => 'INS_EMP_TRAIL', -
                             enable => TRUE );

PL/SQL procedure successfully completed.

CREATE TABLE aud_emp_trail (
    object_schema VARCHAR2(80),
    object_name   VARCHAR2(80),
    policy_name   VARCHAR2(80));

Table created.

CREATE OR REPLACE PROCEDURE system.ins_emp_trail (
    p_object_schema VARCHAR2,
    p_object_name   VARCHAR2,
    p_policy_name   VARCHAR2)
AS
BEGIN
    INSERT INTO system.aud_emp_trail
    VALUES ( p_object_schema, p_object_name, p_policy_name );
END;
/

Procedure created.

CONNECT ALICE/ALICE
Connected.

SELECT * FROM scott.emp;

     EMPNO ENAME                JOB                       MGR HIREDATE         SAL       COMM     DEPTNO
---------- -------------------- ------------------ ---------- --------- ---------- ---------- ----------
      7369 SMITH                CLERK                    7902 17-DEC-80        800                    20
      7499 ALLEN                SALESMAN                 7698 20-FEB-81       1600        300         30
      7521 WARD                 SALESMAN                 7698 22-FEB-81       1250        500         30
      7566 JONES                MANAGER                  7839 02-APR-81       2975                    20
      7654 MARTIN               SALESMAN                 7698 28-SEP-81       1250       1400         30
      7698 BLAKE                MANAGER                  7839 01-MAY-81       2850                    30
      7782 CLARK                MANAGER                  7839 09-JUN-81       2450                    10
      7788 SCOTT                ANALYST                  7566 09-DEC-82       3000                    20
      7839 KING                 PRESIDENT                     17-NOV-81       5000                    10
      7844 TURNER               SALESMAN                 7698 08-SEP-81       1500          0         30
      7876 ADAMS                CLERK                    7788 12-JAN-83       1100                    20
      7900 JAMES                CLERK                    7698 03-DEC-81        950                    30
      7902 FORD                 ANALYST                  7566 03-DEC-81       3000                    20
      7934 MILLER               CLERK                    7782 23-JAN-82       1300                    10

14 rows selected.

-- 모든 데이터가 조회되므로 audit_condition을 만족한다.
```

```
SELECT empno, ename, sal FROM scott.emp;

     EMPNO ENAME                          SAL
---------- -------------------- ----------
      7369 SMITH                         800
      7499 ALLEN                        1600
      7521 WARD                         1250
      7566 JONES                        2975
      7654 MARTIN                       1250
      7698 BLAKE                        2850
      7782 CLARK                        2450
      7788 SCOTT                        3000
      7839 KING                         5000
      7844 TURNER                       1500
      7876 ADAMS                        1100
      7900 JAMES                         950
      7902 FORD                         3000
      7934 MILLER                       1300

14 rows selected.

-- 역시 WHERE 절이 존재하지 않고, 조회하는 컬럼중에 sal이 있으므로 Audit Trail 생성된다.

CONNECT system/manager

Connected.

SELECT ename FROM scott.emp WHERE deptno = 20;

ENAME
--------------------
SMITH
JONES
SCOTT
ADAMS
FORD

-- Audit Column에도 존재하지 않고, Audit Condition에 만족하지 않으므로 Audit Trail이 생성되지 않는다.

SELECT ename, sal, mgr FROM scott.emp WHERE deptno = 30;

ENAME                          SAL        MGR
-------------------- ---------- ----------
ALLEN                         1600       7698
WARD                          1250       7698
MARTIN                        1250       7698
BLAKE                         2850       7839
TURNER                        1500       7698
JAMES                          950       7698

6 rows selected.

-- Audit Column에는 존재하는 WHERE 절에 만족하는 데이터가 Audit Condition에 만족하는 데이터를 포함하지
않으므로 Audit Trail이 생성되지 않는다.

COL db_user FORMAT A10
COL sql_bind FORMAT A40
COL sql_text FORMAT A60
col policy_name for a20
SET LINESIZE 140
```

```
SELECT TO_CHAR(timestamp, 'YYMMDDHH24MI') AS timestamp,
     db_user, policy_name, sql_text ,userhost
FROM dba_fga_audit_trail;

TIMESTAMP       DB_USER    POLICY_NAME          SQL_TEXT                                          USERHOST
--------------- ---------- -------------------- ------------------------------------------------- --------
0707291910      ALICE      AUD_EMP_SAL_MGR      SELECT * FROM scott.emp                             PC1
0707291911      ALICE      AUD_EMP_SAL_MGR      SELECT empno, ename, sal FROM scott.emp             PC1

COLUMN object_schema FORMAT A15
COLUMN object_name FORMAT A15
COLUMN POLICY_NAME FORMAT A15

SELECT * FROM aud_emp_trail;

OBJECT_SCHEMA   OBJECT_NAME     POLICY_NAME
--------------- --------------- ---------------
SCOTT           EMP             AUD_EMP_SAL_MGR
SCOTT           EMP             AUD_EMP_SAL_MGR
```

## 2.4. Oracle Database 10g FGA 예제1 – STATEMENT_TYPE, AUDIT_TRAIL

```
SQL> show parameter audit_trail

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
audit_trail                          string      NONE

SQL> BEGIN
  2      DBMS_FGA.ADD_POLICY(object_schema => 'SCOTT',
  3                          object_name => 'EMP',
  4                          policy_name => 'POL_SCOTT_EMP',
  5                          audit_condition => 'sal > 3000',
  6                          enable => TRUE,
  7                          statement_types => 'SELECT, INSERT',
  8                          audit_trail => DBMS_FGA.DB_EXTENDED);
  9  END;
 10  /

PL/SQL procedure successfully completed.

SQL> SELECT object_schema, object_name, policy_name, policy_text, policy_column, enabled
  2  FROM dba_audit_policies;

OBJECT_SCHEM OBJECT_NAME POLICY_NAME     POLICY_TEXT          POLICY_COLUMN        ENA
------------ ----------- --------------- -------------------- -------------------- ---
SCOTT        EMP         POL_SCOTT_EMP   sal > 3000                                YES

SQL> SELECT timestamp, object_name, scn, sql_text, sql_bind
  2  FROM dba_fga_audit_trail;

no rows selected

SQL> insert into emp values (:empno, :ename, 'MANAGER', 7499, SYSDATE, 3100, '', 20);

1 row created.
```

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE            SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------ ---------- ---------- ----------
      9999 TESTER     MANAGER          7499 05-AUG-07          3100                    20
      7499 ALLEN      SALESMAN         7698 20-FEB-81          1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81          1250        500         30
      7566 JONES      MANAGER          7839 02-APR-81          2975                    20
      7654 MARTIN     SALESMAN         7698 28-SEP-81          1250       1400         30
      7698 BLAKE      MANAGER          7839 01-MAY-81          2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81          2450                    10
      7788 SCOTT      ANALYST          7566 09-DEC-82          3000                    20
      7839 KING       PRESIDENT             17-NOV-81          5000                    10
      7566 JONES      MANAGER          7839 02-APR-81          2975                    20
      7654 MARTIN     SALESMAN         7698 28-SEP-81          1250       1400         30
      7698 BLAKE      MANAGER          7839 01-MAY-81          2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81          2450                    10
      7788 SCOTT      ANALYST          7566 09-DEC-82          3000                    20
      7839 KING       PRESIDENT             17-NOV-81          5000                    10

16 rows selected.

SQL> insert into emp values (8888, 'TESTER2', 'SALESMAN', 7499, SYSDATE, 3200, 10, 30);

1 row created.

SQL> delete from emp where empno = 9999;

1 row deleted.

SQL> select timestamp, object_name, scn, sql_bind, sql_text,
  2  from dba_fga_audit_trail;

TIMESTAMP           OBJECT_NAME          SCN SQL_BIND
------------------- ------------ ---------- --------------------------------
SQL_TEXT
--------------------------------------------------------------------------------
2007-08-05 11:36:21 EMP                   0  #1(4):9999 #2(6):TESTER
insert into emp values (:empno, :ename, 'MANAGER', 7499, SYSDATE, 3100, '', 20)

2007-08-05 11:36:26 EMP          9.2040E+12
select * from emp;

2007-08-05 11:36:34 EMP                   0
insert into emp values (8888, 'TESTER2', 'SALESMAN', 7499, SYSDATE, 3200, 10, 30)
```

* Oracle Database 10g 부터 statement_types과 audit_trail이 추가되었다. Oracle 9i Database에서 SELECT 문만 auditing이 가능했지만 Oracle Database 10g 에서는 INSERT, DELETE, UPDATE, MERGE 에 대해서도 가능하다. 또한 audit_trail 파라메터에 DBMS_FGA.DB_EXTENDED로 설정하므로 init parameter 의 audit_trail 파라메터와 무관하게 FGA 가 설정만으로 수행된 구문과 바인드 정보를 볼 수 있다.

* 위 예제에서 보았듯이 statement_types 가 'SELECT', 'INSERT' 이므로 두가지의 구문형식에 대해서만 Auditing 되며, 'DELETE' 는 Auditing되지 않았다. 그리고 audit_trail 이 DB_EXTENEDED로 설정되어 sql_bind, 와 sql_text에 대해서도 audit trail이 작성되었다.

## 2.5. Oracle Database 10g FGA 예제2 – AUDIT_COLUMN_OPTS

```
SQL> conn / as sysdba
Connected.

SQL> truncate table fga_log$;

Table truncated.

SQL> BEGIN
  2      DBMS_FGA.DROP_POLICY(object_schema => 'SCOTT',
  3                           object_name => 'EMP',
  4                           policy_name => 'POL_SCOTT_EMP');
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2      DBMS_FGA.ADD_POLICY(object_schema => 'SCOTT',
  3                                          object_name => 'EMP',
  4                                          policy_name => 'POL_SCOTT_EMP1',
  5                                          audit_column => 'empno, sal, comm',
  6                                          enable => TRUE,
  7                                          audit_column_opts => DBMS_FGA.ANY_COLUMNS);
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL> select timestamp, object_name, scn, sql_bind, sql_text
  2  from dba_fga_audit_trail;

no rows selected

SQL> connect scott/tiger
Connected.

SQL> select empno, ename from emp where deptno = 20;

     EMPNO ENAME
---------- ----------
      7566 JONES
      7788 SCOTT
      7566 JONES
      7788 SCOTT

SQL> select empno, sal from emp where sal > 3000;

     EMPNO        SAL
---------- ----------
      7839       5000
      7839       5000

SQL> select ename, job from emp where job > 'MANAGER';

ENAME      JOB
---------- ---------
ALLEN      SALESMAN
WARD       SALESMAN
MARTIN     SALESMAN
KING       PRESIDENT
MARTIN     SALESMAN
KING       PRESIDENT
```

```
SQL> select empno, sal, comm from emp where deptno = 30;

     EMPNO        SAL       COMM
---------- ---------- ----------
      7499       1600        300
      7521       1250        500
      7654       1250       1400
      7698       2850
      7654       1250       1400
      7698       2850

SQL> select timestamp, object_name, scn, sql_bind, sql_text
  2  from dba_fga_audit_trail;

TIMESTAMP           OBJECT_NAME                    SCN SQL_BIND
------------------- -------------------- ---------- --------------------------------
SQL_TEXT
--------------------------------------------------------------------------------
2007-08-05 12:57:18 EMP                    9.2040E+12
select empno, ename from emp where deptno = 20

2007-08-05 12:57:32 EMP                    9.2040E+12
select empno, sal from emp where sal > 3000

2007-08-05 12:58:07 EMP                    9.2040E+12
select empno, sal, comm from emp where deptno = 30

* audit_column_opts 값이 DBMS_FGA.ANY_COLUMNS 으로 설정했을 경우 audit_column 에 나열된 Column 중에
1 개라도 포함이 되어 있다면 Audit Trail 이 기록된다. 나열된 컬럼들 중에 없는 경우는 기록되지 않는다.

SQL> truncate table fga_log$;

Table truncated.

SQL> BEGIN
  2      DBMS_FGA.DROP_POLICY(object_schema => 'SCOTT',
  3                           object_name => 'EMP',
  4                           policy_name => 'POL_SCOTT_EMP1');
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2      DBMS_FGA.ADD_POLICY(object_schema => 'SCOTT',
  3                                         object_name => 'EMP',
  4                                         policy_name => 'POL_SCOTT_EMP2',
  5                                         audit_column => 'empno, sal, comm',
  6                                         enable => TRUE,
  7                                         audit_column_opts => DBMS_FGA.ALL_COLUMNS);
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL> connect scott/tiger
Connected.
```

```
SQL> select empno, ename from emp where deptno = 20;

     EMPNO ENAME
---------- ----------
      7566 JONES
      7788 SCOTT
      7566 JONES
      7788 SCOTT

SQL> select empno, sal from emp where sal > 3000;

     EMPNO        SAL
---------- ----------
      7839       5000
      7839       5000

SQL> select ename, job from emp where job > 'MANAGER';

ENAME      JOB
---------- ---------
ALLEN      SALESMAN
WARD       SALESMAN
MARTIN     SALESMAN
KING       PRESIDENT
MARTIN     SALESMAN
KING       PRESIDENT

SQL> select empno, sal, comm from emp where deptno = 30;

     EMPNO        SAL       COMM
---------- ---------- ----------
      7499       1600        300
      7521       1250        500
      7654       1250       1400
      7698       2850
      7654       1250       1400
      7698       2850

SQL> select timestamp, object_name, scn, sql_bind, sql_text
  2  from dba_fga_audit_trail;

TIMESTAMP           OBJECT_NAME                    SCN SQL_BIND
------------------- ---------------------- ---------- --------------------------------
SQL_TEXT
--------------------------------------------------------------------------------
2007-08-05 13:01:07 EMP                     9.2040E+12
select empno, sal, comm from emp where deptno = 30
```

* 위와 같이 audit_column_opts 를 DBMS_FGA.ALL_COLUMN 으로 설정한 경우, audit_column 에 나열한 컬럼이 모두 조회 되는 구문에서만 Audit Trail 이 기록된다.

## 2.6. Database Auditing과 FGA의 비교

이상 살펴 보았듯이 Database Auditing, Trigger를 이용한 Auditing, FGA(Fine-Grained Auditing)을 사용하는 Auditing 기법에 대해 알아보았다. 기존 Database Auditing에서는 그 기능에 제약이 있었지만, Oracle 9i Database 에서 제공되는 FGA를 사용하면 보다 용이한 Auditing 기능을 제공한다. 적은 Audit Trail을 생성하면서 수행되는 SQL 구문에 대한 정보를 얻을 수 있고, Oracle Database 10g 부터 제공되는 FGA를 사용하면 Bind 정보 및 실제 수행된 SQL문의 유용한 정보 등를 얻을 수 있다.

끝으로 변경된 데이터의 이력을 찾아야 하는 업무에서 사전에 적절하게 FGA를 응용한다면 예상치 못한 데이터 유실같은 문제에 대한 원인 규명 및 대처를 할 수 있을 것이라고 기대한다.

# 3. 부록

## 3.1. Trigger를 이용한 DML Auditing – AUDIT_UTIL PACKAGE

```
--Trigger 를 이용한 DML Auditing - Package Name : AUDIT_UTIL

CREATE OR REPLACE PACKAGE AUDIT_UTIL
IS
    -- AUDIT 테이블 생성
    PROCEDURE create_audit_table(
        dest_table_owner     IN VARCHAR2,
        dest_table_name      IN VARCHAR2,
        audit_table_name     IN VARCHAR2 DEFAULT '',
        tablespace_name      IN VARCHAR2 DEFAULT 'SYSTEM');

    -- TRIGGER 생성
    PROCEDURE create_audit_trigger(
        audit_table_name     IN VARCHAR2,
        dest_table_owner     IN VARCHAR2,
        dest_table_name      IN VARCHAR2);

    -- DML AUDIT 를 위한 테이블 및 트리거 자동 생성
    PROCEDURE create_dml_audit(
        dest_table_owner     IN VARCHAR2,
        dest_table_name      IN VARCHAR2,
        audit_table_name     IN VARCHAR2 DEFAULT '',
        tablespace_name      IN VARCHAR2 DEFAULT 'SYSTEM');

    -- DML AUDIT 관련 테이블 및 트리거 제거
    PROCEDURE remove_dml_audit(
        audit_table_name   IN VARCHAR2,
        audit_trigger_name IN VARCHAR2 DEFAULT '');

    -- ADUIT TABLE 내 데이터 삭제 (날짜별 이전 데이터 삭제)
    PROCEDURE delete_auditing_values(
        audit_table_name IN VARCHAR2,
        start_date IN DATE,
        end_date IN DATE DEFAULT NULL);

    -- ADUIT TABLE 내 데이터 삭제 (지정 일수 이전 데이터 삭제)
    PROCEDURE delete_auditing_values(
        audit_table_name IN VARCHAR2,
        days IN NUMBER);
END AUDIT_UTIL;
/

CREATE OR REPLACE PACKAGE BODY AUDIT_UTIL
IS
    -- 0. DDL 실행 프로시져(PRIVATE)
    PROCEDURE exec_ddl(statement IN VARCHAR2)
    IS
        v_statment         VARCHAR2(4000);
        v_cid              INTEGER;
        v_ret              INTEGER;
    BEGIN
        v_statment := REPLACE(statement, CHR(10), '');
        v_cid := DBMS_SQL.OPEN_CURSOR;
        DBMS_SQL.PARSE(C => v_cid,
                       STATEMENT => v_statment,
                       LANGUAGE_FLAG => DBMS_SQL.NATIVE);
```

```
    --v_ret := DBMS_SQL.EXECUTE(v_cid);
    DBMS_SQL.CLOSE_CURSOR(v_cid);

END exec_ddl;

-- 0. DML 실행 프로시져(PRIVATE)
PROCEDURE exec_dml(statement IN VARCHAR2)
IS
    v_statment          VARCHAR2(4000);
    v_cid               INTEGER;
    v_ret               INTEGER;
BEGIN
    v_statment := REPLACE(statement, CHR(10), '');
    v_cid := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(C => v_cid,
                   STATEMENT => v_statment,
                   LANGUAGE_FLAG => DBMS_SQL.NATIVE);
    v_ret := DBMS_SQL.EXECUTE(v_cid);
    DBMS_SQL.CLOSE_CURSOR(v_cid);

END exec_dml;

-- 1. AUDIT 테이블 생성 PROCEDURE : CREATE_AUDIT_TABLE (only SYS OR SYSTEM USER)
PROCEDURE create_audit_table (
    dest_table_owner     in VARCHAR2,
    dest_table_name      in VARCHAR2,
    audit_table_name     in VARCHAR2 DEFAULT '',
    tablespace_name      in VARCHAR2 DEFAULT 'SYSTEM')
IS
    CURSOR c_table_columns IS
        SELECT column_name, data_type, data_length, data_precision, data_scale
        FROM dba_tab_columns
        WHERE owner = dest_table_owner
        AND table_name = dest_table_name;

    v_ddl               VARCHAR2(4000);
    v_audit_table_name VARCHAR2(100);
BEGIN
    IF (dest_table_owner IS NULL ) OR (dest_table_name IS NULL) THEN
      DBMS_OUTPUT.PUT_LINE('No Values in Owner or Table_name');
    END IF;

    IF (audit_table_name IS NULL) THEN
      v_audit_table_name := dest_table_name || '_AUDIT';
    ELSE
      v_audit_table_name := audit_table_name;
    END IF;
    -- Table DDL 문
    v_ddl := 'CREATE TABLE ' || v_audit_table_name || ' (' || chr(10);

    FOR v_col_rec IN c_table_columns LOOP
        IF (c_table_columns%NOTFOUND) THEN
          DBMS_OUTPUT.PUT_LINE('TABLE NOT FOUND');
        END IF;

     CASE
         WHEN v_col_rec.data_type IN ('VARCHAR2', 'NVARCHAR2', 'CHAR', 'NCHAR') THEN
             v_ddl := v_ddl || '    old_' || v_col_rec.column_name || ' '
                    || v_col_rec.data_type ||'(' || v_col_rec.data_length || ')';
         WHEN v_col_rec.data_type = 'NUMBER' THEN
```

```
            v_ddl := v_ddl || '    old_' || v_col_rec.column_name || ' ' || v_col_rec.data_type;
            IF (v_col_rec.data_precision IS NOT NULL) THEN
                v_ddl := v_ddl || '(' || v_col_rec.data_precision;
                IF (v_col_rec.data_scale IS NOT NULL) THEN
                    v_ddl := v_ddl || ',' || v_col_rec.data_scale;
                END IF;
                v_ddl := v_ddl || ')';
            END IF;
        ELSE
            v_ddl := v_ddl || '    old_' || v_col_rec.column_name || ' ' || v_col_rec.data_type;
        END CASE;
     v_ddl := v_ddl || ',' || CHR(10);
    END LOOP;

    FOR v_col_rec IN c_table_columns LOOP
        IF (c_table_columns%NOTFOUND) THEN
          DBMS_OUTPUT.PUT_LINE('TABLE NOT FOUND');
        END IF;

     CASE
         WHEN v_col_rec.data_type IN ('VARCHAR2', 'NVARCHAR2', 'CHAR', 'NCHAR') THEN
            v_ddl := v_ddl || '    new_' || v_col_rec.column_name || ' '
                    || v_col_rec.data_type ||'(' || v_col_rec.data_length || ')';
         WHEN v_col_rec.data_type = 'NUMBER' THEN
            v_ddl := v_ddl || '    new_' || v_col_rec.column_name || ' ' || v_col_rec.data_type;
            IF (v_col_rec.data_precision IS NOT NULL) THEN
                v_ddl := v_ddl || '(' || v_col_rec.data_precision;
                IF (v_col_rec.data_scale IS NOT NULL) THEN
                    v_ddl := v_ddl || ',' || v_col_rec.data_scale;
                END IF;
                v_ddl := v_ddl || ')';
            END IF;
         ELSE
            v_ddl := v_ddl || '    new_' || v_col_rec.column_name || ' ' || v_col_rec.data_type;
        END CASE;
     v_ddl := v_ddl || ',' || CHR(10);
    END LOOP;

    v_ddl := v_ddl || '    changed_by VARCHAR2(8),' || CHR(10);
    v_ddl := v_ddl || '    change_type CHAR(1),' || CHR(10);
    v_ddl := v_ddl || '    timestamp DATE' || CHR(10) || ') TABLESPACE ' || tablespace_name;

    exec_ddl(v_ddl);

END create_audit_table;

-- 2. TRIGGER 생성 PROCEDURE : CREATE_AUDIT_TRIGGER
PROCEDURE create_audit_trigger(
    audit_table_name  IN VARCHAR2,
    dest_table_owner  IN VARCHAR2,
    dest_table_name   IN VARCHAR2)
IS
    CURSOR c_table_columns IS
        SELECT rownum, column_name FROM dba_tab_columns
        WHERE owner = USER
        AND table_name = audit_table_name;

    v_audit_table_name VARCHAR2(100);
    v_st  VARCHAR2(4000);
BEGIN
```

```
        v_audit_table_name := audit_table_name;
        v_st := 'CREATE OR REPLACE TRIGGER TR_' || v_audit_table_name|| CHR(10)
              || '    BEFORE INSERT OR DELETE OR UPDATE ON ' || dest_table_owner || '.' ||
dest_table_name || CHR(10)
              || '    FOR EACH ROW '|| CHR(10)
              || 'DECLARE' || CHR(10)
              || '    v_ChangeType CHAR(1); ' || CHR(10)
              || 'BEGIN ' || CHR(10)
              || '    IF INSERTING THEN v_ChangeType := ''I''; ' || CHR(10)
              || '    ELSIF UPDATING THEN v_ChangeType := ''U''; ' || CHR(10)
              || '    ELSE v_ChangeType := ''D''; ' || CHR(10)
              || '    END IF; ' || CHR(10)
              || '    INSERT INTO ' || v_audit_table_name || '(';

        FOR v_col_rec IN c_table_columns LOOP
            IF (v_col_rec.rownum > 1) THEN
                v_st := v_st || ', ';
            END IF;
            v_st := v_st || v_col_rec.column_name;
        END LOOP;

        v_st := v_st || ') ' || CHR(10) || '    VALUES (';
        FOR v_col_rec IN c_table_columns LOOP
            IF (v_col_rec.rownum > 1) THEN
                v_st := v_st || ', ';
            END IF;
            CASE SUBSTR(v_col_rec.column_name, 1, 3)
                WHEN 'OLD' THEN
                    v_st := v_st || REPLACE(v_col_rec.column_name, 'OLD_', ':OLD.');
                WHEN 'NEW' THEN
                    v_st := v_st || REPLACE(v_col_rec.column_name, 'NEW_', ':NEW.');
                ELSE
                   IF (v_col_rec.column_name = 'CHANGED_BY') THEN
                       v_st := v_st || 'USER';
                   ELSIF (v_col_rec.column_name = 'CHANGE_TYPE') THEN
                       v_st := v_st || 'v_ChangeType';
                   ELSIF (v_col_rec.column_name = 'TIMESTAMP') THEN
                       v_st := v_st || 'SYSDATE';
                   END IF;
            END CASE;
        END LOOP;

        v_st := v_st || '); ' || CHR(10)
             || 'END TR_' || v_audit_table_name || ';';


        exec_ddl(v_st);

    END create_audit_trigger;

    -- 3. DML AUDIT 를 위한 테이블 및 트리거 자동 생성 PROCEDURE : CREATE_DML_AUDIT
    PROCEDURE create_dml_audit(
        dest_table_owner     in VARCHAR2,
        dest_table_name      in VARCHAR2,
        audit_table_name     in VARCHAR2 DEFAULT '',
        tablespace_name      in VARCHAR2 DEFAULT 'SYSTEM')
    IS
        v_audit_table_name VARCHAR2(100);
        v_dest_table_owner VARCHAR2(100);
        V_dest_table_name VARCHAR2(100);
```

```
BEGIN
    v_dest_table_owner := UPPER(dest_table_owner);
    v_dest_table_name := UPPER(dest_table_name);

    IF (audit_table_name IS NULL) THEN
        v_audit_table_name := UPPER(dest_table_name) || '_AUDIT';
    ELSE
        v_audit_table_name := UPPER(audit_table_name);
    END IF;

   CREATE_AUDIT_TABLE(v_dest_table_owner, v_dest_table_name, v_audit_table_name, tablespace_name);
   CREATE_AUDIT_TRIGGER(v_audit_table_name, v_dest_table_owner, v_dest_table_name);
END create_dml_audit;

-- 4. DML AUDIT 관련 테이블 및 트리거 제거 PROCEDURE : MAKE_DML_AUDIT
PROCEDURE remove_dml_audit(
    audit_table_name   IN VARCHAR2,
    audit_trigger_name IN VARCHAR2 DEFAULT '')
IS
    v_c  INTEGER;
    v_st VARCHAR2(4000);
    v_tr VARCHAR2(100);
BEGIN
    v_st := 'DROP TABLE ' || audit_table_name;
    exec_ddl(v_st);
    IF (audit_trigger_name IS NULL) THEN
      v_tr := 'TR_' || audit_table_name;
    ELSE
      v_tr := audit_trigger_name;
    END IF;
    v_st := 'DROP TRIGGER ' || v_tr;
    exec_ddl(v_st);
END remove_dml_audit;

-- 5. ADUIT TABLE 내 데이터 삭제 PROCEDURE : DELETE_AUDITING_VALUES
PROCEDURE delete_auditing_values(
    audit_table_name IN VARCHAR2,
    start_date IN DATE,
    end_date IN DATE DEFAULT NULL)
IS
    v_st VARCHAR2(4000);
BEGIN
    v_st := 'DELETE FROM' || audit_table_name;
    IF (end_date IS NULL) THEN
        v_st := v_st || ' WHERE TIMESTAMP < TO_DATE('''
             || TO_CHAR(start_date, 'YYYY-MM-DD HH24:MI:SS')
             || ''', ''fmYYYY-MM-DD HH24:MI:SS'')';
    ELSE
        v_st := v_st || ' WHERE TIMESTAMP BETWEEN TO_DATE('''
             || TO_CHAR(start_date, 'YYYY-MM-DD HH24:MI:SS')
             || ''', ''fmYYYY-MM-DD HH24:MI:SS'') AND TO_DATE('''
             || TO_CHAR(end_date, 'fmYYYY-MM-DD HH24:MI:SS')
             || ''', ''fmYYYY-MM-DD HH24:MI:SS'')';
    END IF;
    exec_dml(v_st);
    COMMIT;
END delete_auditing_values;

PROCEDURE delete_auditing_values(
    audit_table_name IN VARCHAR2,
```

```
        days IN NUMBER)
    IS
        v_st VARCHAR2(4000);
        v_last_date DATE;
    BEGIN
        v_last_date := SYSDATE - days;
        v_st := 'DELETE FROM' || audit_table_name
             || ' WHERE TIMESTAMP < TO_DATE('''
             || TO_CHAR(v_last_date, 'YYYY-MM-DD HH24:MI:SS')
             || ''', ''fmYYYY-MM-DD HH24:MI:SS'')';
        exec_dml(v_st);
        COMMIT;
    END delete_auditing_values;

END AUDIT_UTIL;
/

-------------------------------------------------------------------------------------------------------------
1. 실행방법 ( AUDIT 용 TABLE 및 TRIGGER 생성)
   exec make_dml_audit (dest_table_owner, dest_table_name, W
   audit_table_name(default ''), tablespace_name(default 'SYSTEM');
   EXAMPLE 1
   - exec make_dml_audit('SCOTT', 'EMP');
   EXAMPLE 2
   - exec make_dml_audit('SCOTT', 'EMP', 'EMP_AUDIT', 'USERS');

   확인 : DML 작업 이후 audit_table_name 조회
     EXAMPLE - SELECT * FROM EMP_AUDIT;

2. AUDIT TABLE 내 데이터 삭제 PROCEDURE delete
   EXAMPLE 1
   - exec delete_auditing_values('EMP_AUDIT', TO_DATE('2007-08-01 18:00:00', 'fmYYYY-MM-DD
HH24:MI:SS');
   EXAMPLE 2
   - exec delete_auditing_values('EMP_AUDIT', TO_DATE('2007-08-01 00:00:00', 'fmYYYY-MM-DD
HH24:MI:SS'), TO_DATE('2007-08-01 18:00:00', 'fmYYYY-MM-DD HH24:MI:SS'));
   EXAMPLE 3
   - exec delete_auditing_values('EMP_AUDIT', 1); -- 1일 이전 데이터 삭제
3. 제거방법 ( AUDIT 용 TABLE 및 TRIGGER 제거)
   exec remove_dml_audit (audit_table_name, audit_trigger_name(default ''));
   EXAMPLE - exec remove_dml_audit('EMP_AUDIT');

-- Test : SCOTT.EMP 에 대한 DML Auditing
exec make_dml_audit('SCOTT', 'EMP', 'EMP_AUDIT', 'USERS');

select table_name from dba_tables where trigger_name like '%_AUDIT'
select trigger_name from dba_triggers where trigger_name like 'TR_%'

connect scott/tiger

INSERT INTO emp VALUES ( 9999, 'TESTER', 'CLERK', 7782, SYSDATE, 1000, 0, 10);
COMMIT;
UPDATE emp SET comm = 300 WHERE empno = 9999;
COMMIT;
DELETE FROM emp WHERE empno = 9999;
COMMIT;

connect /as sysdba

SELECT OLD_EMPNO, OLD_ENAME, OLD_COMM, NEW_EMPNO, NEW_ENAME, NEW_COMM, CHANGED_BY, CHANGE_TYPE,
```

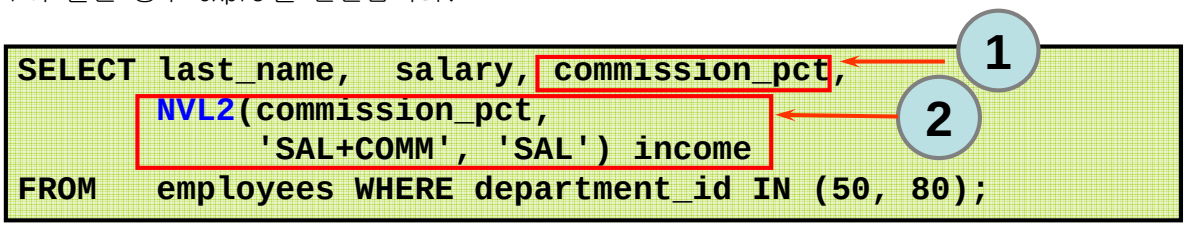
```
TIMESTAMP FROM EMP_AUDIT;

 OLD_EMPNO OLD_ENAME     OLD_COMM  NEW_EMPNO NEW_ENAME     NEW_COMM CHANGED_BY        CH TIMESTAMP
---------- ---------- ---------- ---------- ---------- ---------- ---------------- -- ---------
                                        9999 TESTER              0 SCOTT            I  01-AUG-07
      9999 TESTER               0       9999 TESTER            300 SCOTT            U  01-AUG-07
      9999 TESTER             300                                  SCOTT            D  01-AUG-07

-- 일정 기간에 대한 Audit Trail Reocords 삭제
alter session set nls_date_format = 'yyyy-mm-dd hh24:mi:ss';
exec delete_auditing_values('EMP_AUDIT', TO_DATE('2007-08-01 18:00:00', 'fmYYYY-MM-DD HH24:MI:SS'));

exec delete_auditing_values('EMP_AUDIT', TO_DATE('2007-08-01 00:00:00', 'fmYYYY-MM-DD HH24:MI:SS'),
TO_DATE('2007-08-01 18:00:00', 'fmYYYY-MM-DD HH24:MI:SS'));

exec delete_auditing_values('EMP_AUDIT', 0); -- 현재 이전 삭제(전체 삭제)

-- dbms_job 을 이용하여 자동 삭제 스케줄링
 VARIABLE job_no NUMBER;
VARIABLE inst_no NUMBER;
begin
  select instance_number into :inst_no from v$instance;
  dbms_job.submit(:job_no, 'AUDIT_UTIL(''EMP_AUDIT'', 0);', trunc(sysdate+1,'HH'),
'trunc(SYSDATE+1,''HH'')', TRUE, :inst_no);
  commit;
end;
/
print :job_no;

exec dbms_job.broken(:job_no, TRUE);
exec dbms_job.remove(:job_no);

-- Audit Trail Table 제거
exec remove_dml_audit('EMP_AUDIT');
```

## 3.2. SQL Tips for Developer

이번 호부터 개발자를 위한 SQL Tip 을 연재합니다. 자주 사용하는 Oracle Function 이나, 새로운 오라클 버전에서 소개된 SQL New Feature 를 통하여 프로그램 생산성 및 성능을 높일 수 있는 내용들을 소개하겠습니다. 이번 호에서는 개발자 분들이 자주 사용하는 NVL 함수 관련 내용을 모아 보았습니다. 흔히 개발자 분들이 NULL 값의 처리에 많은 고민을 하게 되는데, 기존에 자주 사용하시는 NVL 함수 뿐 아니라, NVL2, NULLIF, COALESCE, LNNVL 함수를 추가로 소개합니다. 이 함수들은 기존에 프로그램에서 로직으로 처리해야 하는 많은 코딩을 함수 하나를 제대로 사용함으로써 대체할 수 있는 기능을 가지고 있습니다. 프로그램 개발 생산성을 올릴 수 있는 오라클 함수들입니다.

### 1. NVL2 (expr1, expr2, expr3)

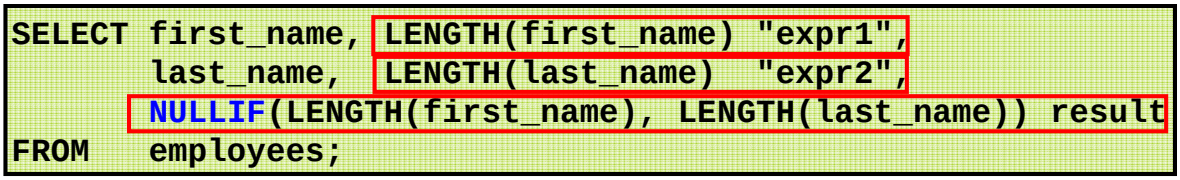
NVL2 함수는 첫번째 표현식을 검사합니다. expr1 이 널이 아닌 경우, NVL2 함수는 expr2 를 반환하며, expr1 이 널인 경우 expr3 을 반환합니다.




예제에서는 COMMISSION_PCT 열을 검사하여 이 값이 NULLL 이 아니면 두번째 표현식인 SAL+COMM 이 반환됩니다. COMMISSION_PCT 열에 널 값이 있으면 세번째 표현식인 SAL 이 반환됩니다.
인수 *expr1* 에는 모든 데이터 유형을 사용할 수 있습니다. 인수 *expr2* 및 *expr3* 에는 LONG 을 제외한 모든 데이터 유형을 사용할 수 있습니다. *expr2* 및 *expr3* 의 데이터 유형이 서로 다를 경우, Oracle server 는 *expr3* 이 널 상수가 아니라면 두 표현식을 비교하기 전에 *expr3* 을 *expr2* 의 데이터 유형으로 변환합니다. *expr3* 이 널 상수인 경우에는 데이터 유형 변환이 필요없습니다. 반환 값의 데이터 유형은 항상 *expr2* 의 데이터 유형과 동일하며 *expr2* 가 문자 데이터인 경우에는 반환 값의 데이터 유형이 VARCHAR2 입니다.

### 2. NULLIF (expr1, expr2)

NULLIF 함수는 두 표현식을 비교하여 동일한 경우 널을 반환하고 동일하지 않은 경우 첫번째 표현식을 반환합니다. 첫번째 표현식에 널 리터럴을 지정할 수 없습니다.



예제에서는 EMPLOYEES 테이블의 First_name 과 Last_name 의 길이를 비교하여 같으면 NULL 을 리턴하고 다르면 First_name 의 길이를 리턴합니다.

**참고:** NULLIF 함수는 다음의 CASE 표현식과 논리적으로 동일합니다. CASE 표현식은 다음 페이지에서 설명합니다. **[ CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END ]**

### 3. COALESCE (expr1, expr2, ..., expr*n*)

COALESCE 함수는 여러 개의 평가식에 NULL 값이 있는 지 여부를 동시에 수행할 수 있습니다. expr1 을 먼저 평가하여 Null 이 아니면 expr1 의 값을 리턴하고 NULL 이면 expr2 를 평가합니다. expr2 가 NULL 이 아니면 expr2 의 값을 리턴하고 NULL 이면 expr3 을 평가합니다. 이런 방식을 expr*n* 까지 모두 평가할 수 있습니다.

```
SELECT   last_name,
         COALESCE(commission_pct, salary, 10) comm
FROM     employees
ORDER BY commission_pct;
```

예제에서, COMMISSION_PCT 값이 널이 아닌 경우, 이 값이 표시됩니다. COMMISSION_PCT 값이 널인 경우, SALARY 가 평가됩니다. COMMISSION_PCT 와 SALARY 값이 모두 널인 경우, 상수 값 10 이 표시됩니다.

### 4. LNNVL (expr1)

LNNVL 함수는 조건식에 주어지는 항의 한쪽 혹은 양쪽에 NULL 값이 나타날 경우에, 조건식을 평가하는 간결한 방법을 제공한다. LNNVL 함수는 where 절에만 사용할 수 있습니다.

```
1. emp 테이블에서 commission 이 500 보다 작은 사람들의 수를 계산한다.
SQL> SELECT COUNT (*) FROM emp WHERE comm < 500
       COUNT(*)
     ----------
             2

2. 하지만, commission 을 못받는 사람을 포함해서(comm is null),
   commissin 이 500 보다 작은 사람들의 수를 계산하기 위해서는
   아래와 같이 LNNVL 함수를 쓴다.

SQL> SELECT COUNT (*) FROM emp WHERE LNNVL(comm >= 500)

       COUNT(*)
     ----------
             12
```

LNNVL 함수는 조건식을 파라미터로 가지며, 조건식이 *FALSE 나 UNKNOWN 일 경우에 TRUE* 를 반환하고, 조건식이 *TRUE 일 경우에 FALSE* 를 반환합니다. LNNVL 함수는 잠재적으로 NULL 값이 나올 수 있는 상황에서 NULL 값을 처리하기 위해 사용할 수 있습니다.

## 3.3. Oracle 소식

### *Oracle 11g 가 출시되었습니다.*

데이터베이스(DB) 업계 최강자인 오라클이 7 월 12 일 차세대 데이터베이스관리시스템(DBMS)인 '오라클 데이터베이스 11g'를 공식 출시하였습니다. 이전 버전인 '오라클 데이터베이스 10g'가 출시된 지 3 년여 만에 선보인 11g 는 리얼 애플리케이션 테스팅 · 재해복구 · 정보순환주기관리 등 400 가지 이상의 기능을 갖췄으며 1500 만시간의 테스팅과 3 만 6000 개월에 걸친 작업으로 개발되었습니다. 특히 자가 관리와 자동화 성능으로 기업이 서비스레벨협약(SLA)를 실현하도록 지원하는 리얼 애플리케이션 테스트 기능은 이 제품이 내세우는 가장 큰 장점입니다.

**◆ 고가용성, 확장성, IT 비용절감 강조**

오라클은 향상된 보안기능과 그리드 컴퓨팅 기반 고가용성, 확장성을 내세워 '오라클 데이터베이스 11g'를 사용하는 기업이 시스템 관리 비용과 데이터 스토리지 비용을 절감할 수 있다는 것을 강조하고 있습니다.

서버 테크놀러지 부문 앤드류 멘델슨 수석부사장은 컨퍼런스콜을 통해 "오라클 데이터베이스 11g 는 그리드 기술이 확장돼 기업의 IT 비용절감에 도움을 준다"고 말하고 있습니다. '오라클 데이터베이스 11g'가 기업의 비용절감을 돕는다는 점을 강조하고 있으며 앞으로 다양한 운영체제(OS)에 대해 동일한 수준의 지원을 펼칠 계획입니다. 또 오라클은 최근 비즈니스인텔리전스(BI)에 대한 기업 수요가 늘어난다는 것을 고려, 신제품의 BI 기능도 강화하고 있습니다.

**◆ 국내에서의 반응은?**

'오라클 데이터베이스 11g'는 국내 시장에서는 오는 8 월에 출시될 예정입니다.
그러나, 현재 대부분의 기업이 '오라클 데이터베이스 10g'보다 이전 버전을 사용하고 있고, 국내 기업들이 중요한 데이터 집합소인 DB 를 업그레이드하거나 교체하는 데 민감한 반응을 보이고 있기 때문에 신제품 확산 빠르게 적용되지는 않을 것으로 예측됩니다.

한국오라클 세일즈컨설팅 양수환 전무는 "자체 조사 결과 오라클 고객 가운데 40%가 '오라클 데이터베이스 10g' 버전을 사용하고 있다"며 "한국 기업들이 새로운 제품을 받아들이는 속도가 늦긴 하지만 새로운 기능에 대한 요구가 있어 이 제품이 확산되는데 큰 장애가 없다고 본다"고 발표했습니다.

'오라클 데이터베이스 11g'가 이전 버전인 '오라클데이터베이스 10g'와 기능면에서 크게 다르지 않다는 것도 기업들이 새로운 제품을 선택하는데 걸림돌이 될 것으로 보입니다.

실제로 오라클은 '오라클 데이터베이스 11g'에 새로운 기능을 추가시켰다기보다 기능을 확장하고 여러 옵션들을 새롭게 내놓는 것에 중점을 두었으며, 이 신제품이 기업의 IT 비용절감에 도움이 된다고 하지만 이 특징이 기업들이 업그레이드를 강행할 충분한 요소가 될 수 있을 지는 미지수입니다. 한국오라클은 이에 "10g 사용자가 오라클 데이터베이스 11g 로 업그레이드 할 경우 비용이 거의 들지 않기 때문에 업그레이드를 채택하는 기업이 늘어날 것"이라고 합니다.

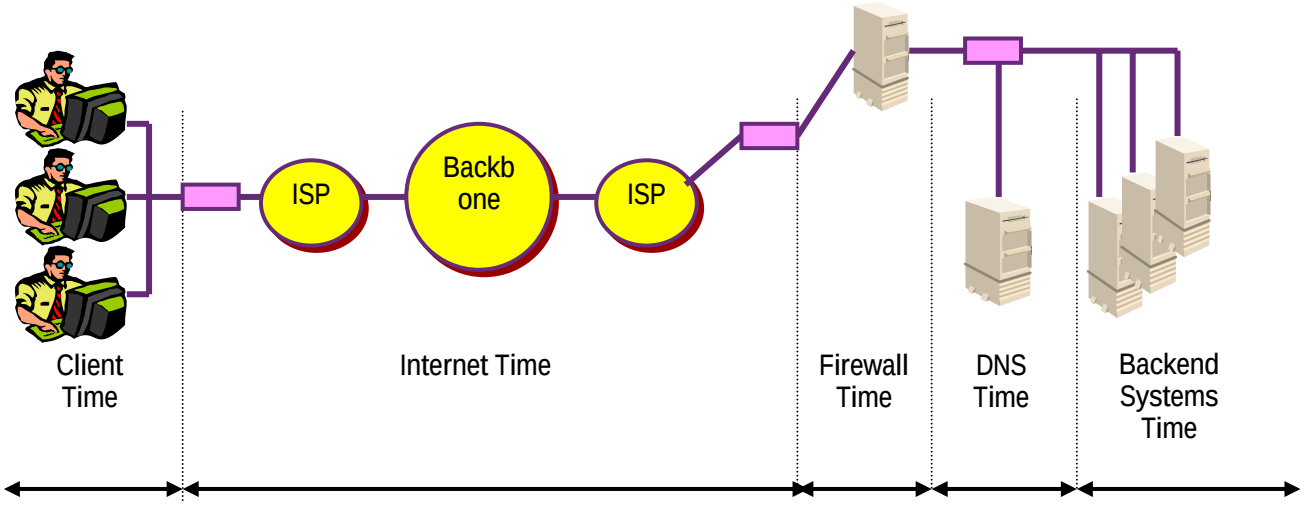
**◆11g 관련 기술 세미나 개최**

한국오라클 교육센터에서는 8 월 22 일부터 24 일까지 3 일간에 걸쳐서 Oracle RDBMS 11g 의 신기술 소개와 RAC(Real Application Cluster), 대용량 데이터 Migration 기법에 대한 세미나를 실시합니다. '2007 Innovative Technology Seminar Track' 을 통해, 오라클 신기술에 대한 핵심적인 내용을 신기술에 대한 접근방법 설명과 데모를 통하여 소개해 드립니다. 오라클 교육사업본부의 전문성을 가장 극대화하여 보여줄 수 있는 분야인, Technology 부분의 주요 3 가지 주제에 대해 세미나를 진행합니다. 'Oracle Database 11g New Features', 'Real Application Cluster 10g' 그리고 'Down-Time 최소화를 위한 다양한 Data Migration 방법론' 3 가지 주제와 일반 Overview 세미나에서는 접하기 힘든 고급 실무 사례도 다룰 예정입니다.

# 4. Pro-Active Tuning Service

## 4.1. 실제 사용자(End-User) 관점의 응답시간 튜닝

Pro-active tuning service 는 사용자 관점의 모니터링 및 분석을 통하여 실제 End-User 가 느끼는 응답시간(Response Time)을 튜닝합니다. APM(Application Performance Management) 툴을 이용하여 End-User 의 Request 결과를 반환받기까지의 모든 구간(Client PC, Internet 구간, FireWall, DNS, Web Server, WAS, DBMS) 을 분석하여 가장 Delay Time 이 많이 소요된 구간을 찾아 냅니다.




## 4.2. 최상의 성능 상태로 비즈니스 고가용성을 유지

**Pro-Active Tuning Service**

- 매 업무 단위 프로젝트 마다 참여하여 업무 적용(Open) 前 문제 요소를 분석하여 튜닝.
- 단위 업무 적용(Open) 후 매 3 개월(데이터량 갱신 주기) 마다 튜닝 포인트를 설정, 성능 둔화 요소를 해결.
- 전사적으로 새롭게 추가되는 업무 단위 프로젝트의 모든 SQL 쿼리를 검토 및 튜닝.
- 다양한 대용량 데이터베이스 관리/튜닝 기법을 도입하여 최적의 DB 상태를 1 년 내내 상시 유지.
- 전략적 튜닝 Factor 를 분석, 투자 대비 효율이 높은 Targeting 기법 적용. (비중도 높은 SQL 을 튜닝함)


**[Pro-Active Tuning Service Schedule]**

*Performance Drop Point 마다 적절한 튜닝을 실시함으로써 항상 최적의 성능을 유지한다. !!*

## 4.3. Knowledge Transfer

Pro-Active Tuning Service 는 고객의 Business Process 를 이해하고 시스템을 분석한 후 튜닝하는 것으로 완료되지 않습니다. 실제로 고객사 환경에서 튜닝한 내용을 그대로 실무자들에게 전수하여 내부 임직원의 역량을 제고시킵니다. 또한, Oracle RDBMS 신 버젼의 New Features 를 교육함으로써, 이용자(관리자 및 개발자)가 스스로 개발 업무의 효율 및 생산성을 향상시킬 수 있도록 지원합니다. 이외에도 DBMS 관리자를 위한 관리 노하우(고급 Trouble-Shooting, 대용량 DB 처리, 병렬 처리 등)를 전수함으로써, 최상의 시스템을 최고의 기술로 유지할 수 있도록 지원합니다.

### UAS (User Adapted Seminar) 진행 사례 및 내용 (Contents)

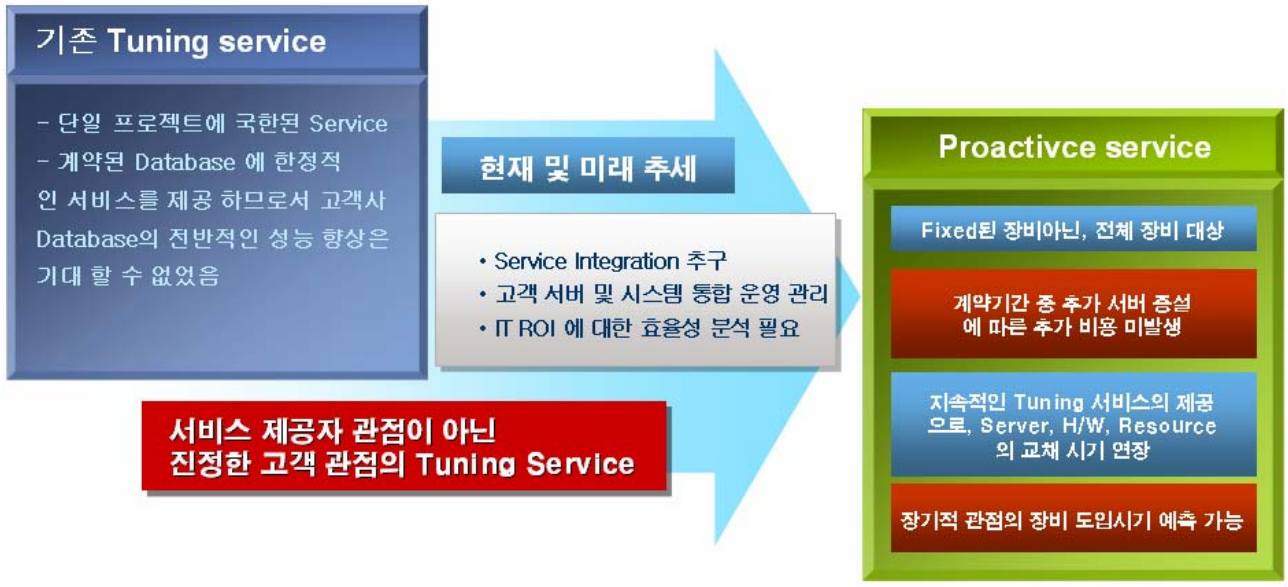
- **개발자를 위한 SQL 튜닝 실무 사례 세미나**
  G 쇼핑몰 업체 튜닝 후 실제 고객사의 튜닝 사례를 개발자들에게 전수하여 개발자들이 성능을 고려한 SQL 을 작성할 수 있도록 내부 역량을 제고시킴.

- **Oracle 10g New Features 세미나**
  S Global 전자 기업 : Oracle 10g 버전으로 업그레이드 하기 전, 신 버전의 새로운 기능과 주의 사항을 전파함으로써, 업그레이드 후 발생할 수 있는 문제점의 사전 제거와 개발자들이 새로운 기능을 이용함으로써, 개발 생산성을 향상시킴.

- **K 국가기관 DBMS 관리 노하우 세미나**
  내부 관리자 (DBA,SE)들을 대상으로 DBMS 관리자들이 흔히 겪을 수 있는 상황에 대한 Administration Know-How 와 고급 Trouble-Shooting 사례를 소개함으로써, 관리 기술력을 향상시킴.

## 4.4. Tuning 범위 확대

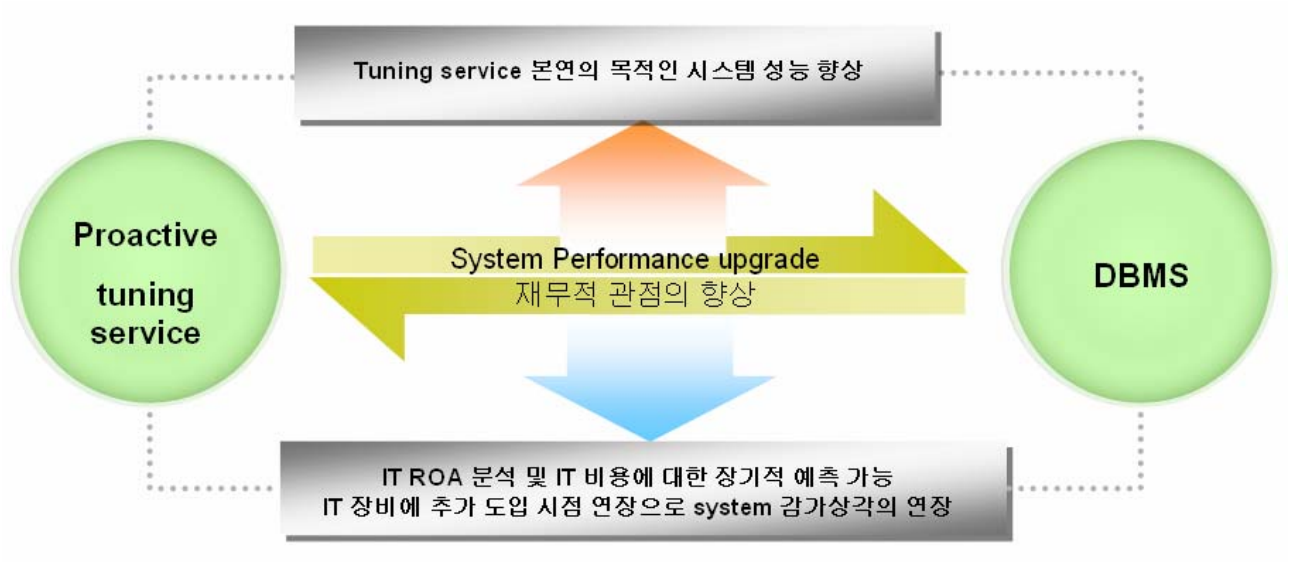
Pro-active tuning service 의 제공 범위는 제한된 Database, 제한된 Server 에 국한 되지 않으며, 고객사 전체의 Server+DB 를 대상으로 그 범위를 확대 함으로써 고객사 전체 Performance 향상에 기여 합니다.



## 4.5. 기대효과

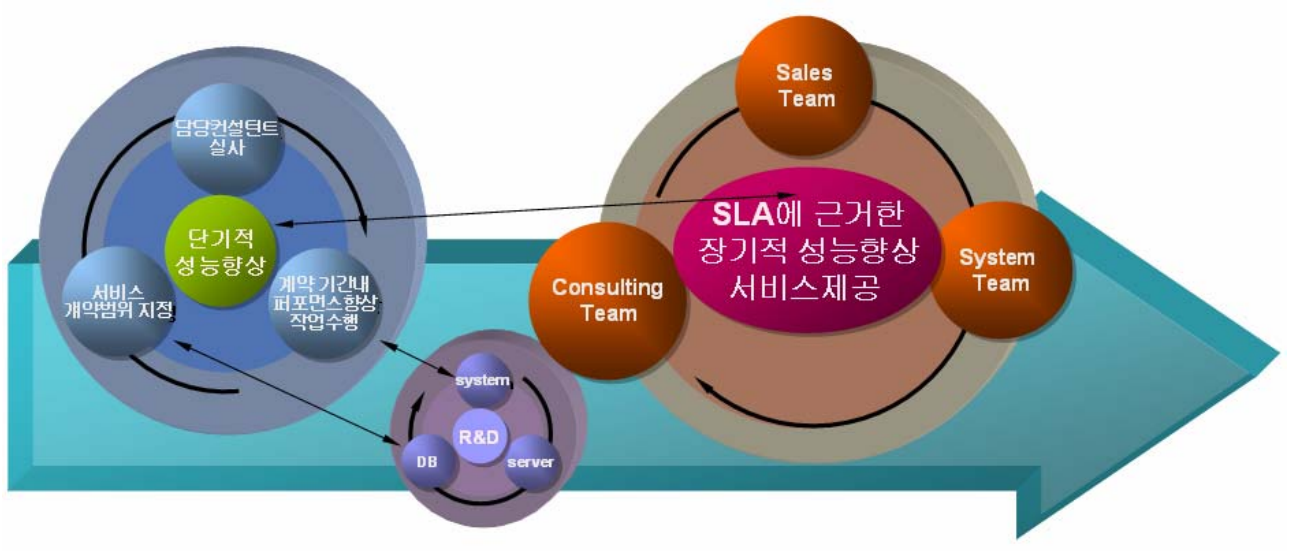
### 4.5.1. 재무적 관점

기존 Tuning Service 는 주로 System Performance 향상에 따른 업무 트레픽 감소에 초점이 맞춰져 있었습니다. Pro-actvice 서비스는 Tuning 작업을 통한 업무 처리 시간 단축 뿐만 아니라, 업무 처리 시간 단축으로 가져 올 수 있는 재무적 성과를 가능하게 합니다.



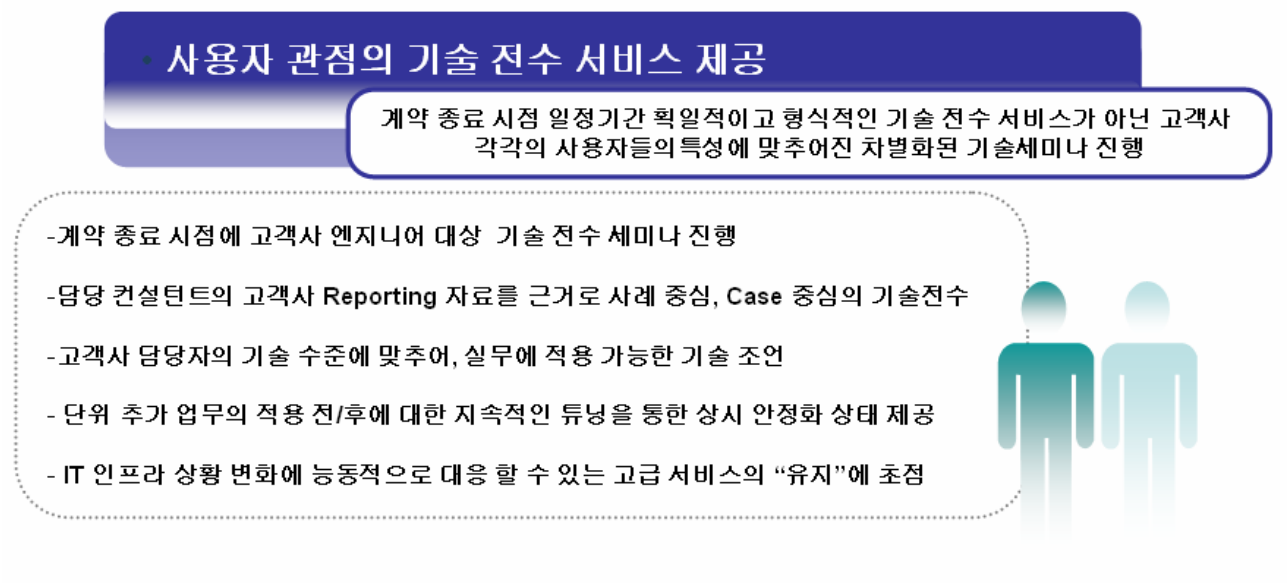

### 4.5.2. 서비스 관점

단기적 성능 향상에 맞추어진 기존 Tuning 서비스는 계약된 system 및 Database 를 서비스 대상으로 하기 때문에 전사적인 차원의 성능 향상을 기대 하기 어려웠습니다. Proactive tuning service 는 계약 기간 동안 주요 비즈니스 Factor 별로 SLA 를 정하여 Tuning consulting 을 수행함으로써 서비스 자체의 안정성을 제고 할 수 있습니다.

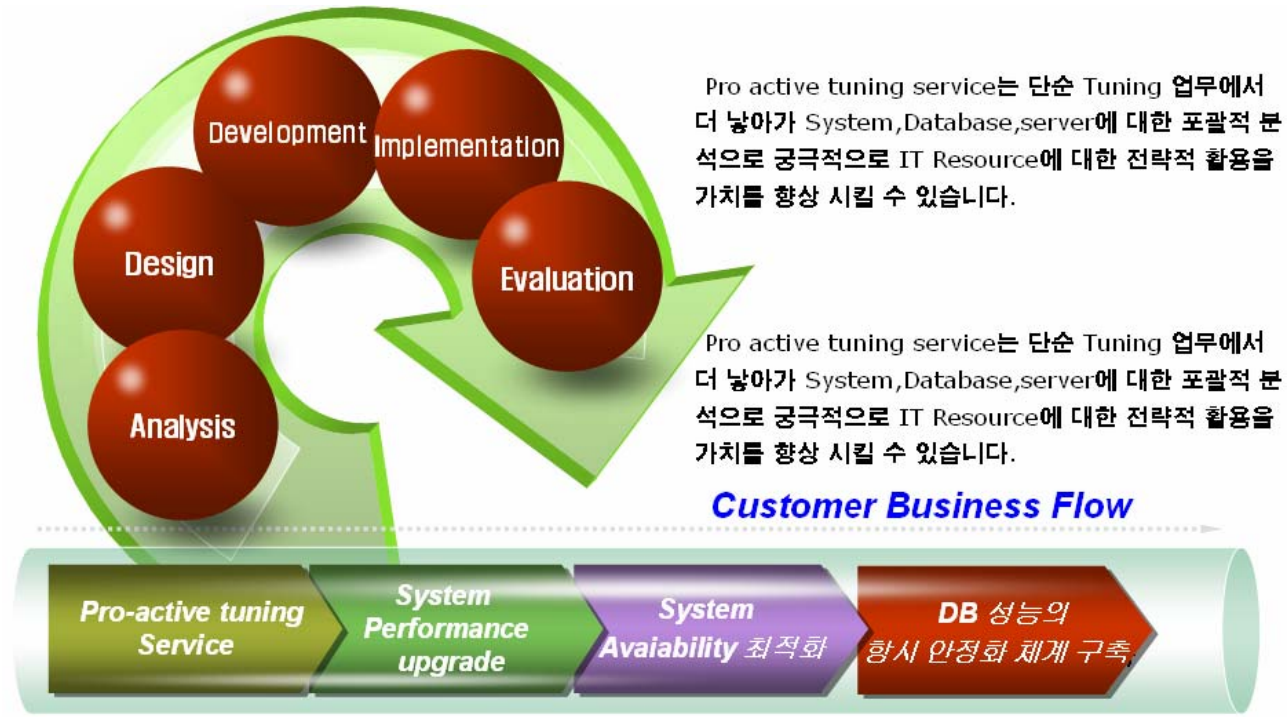


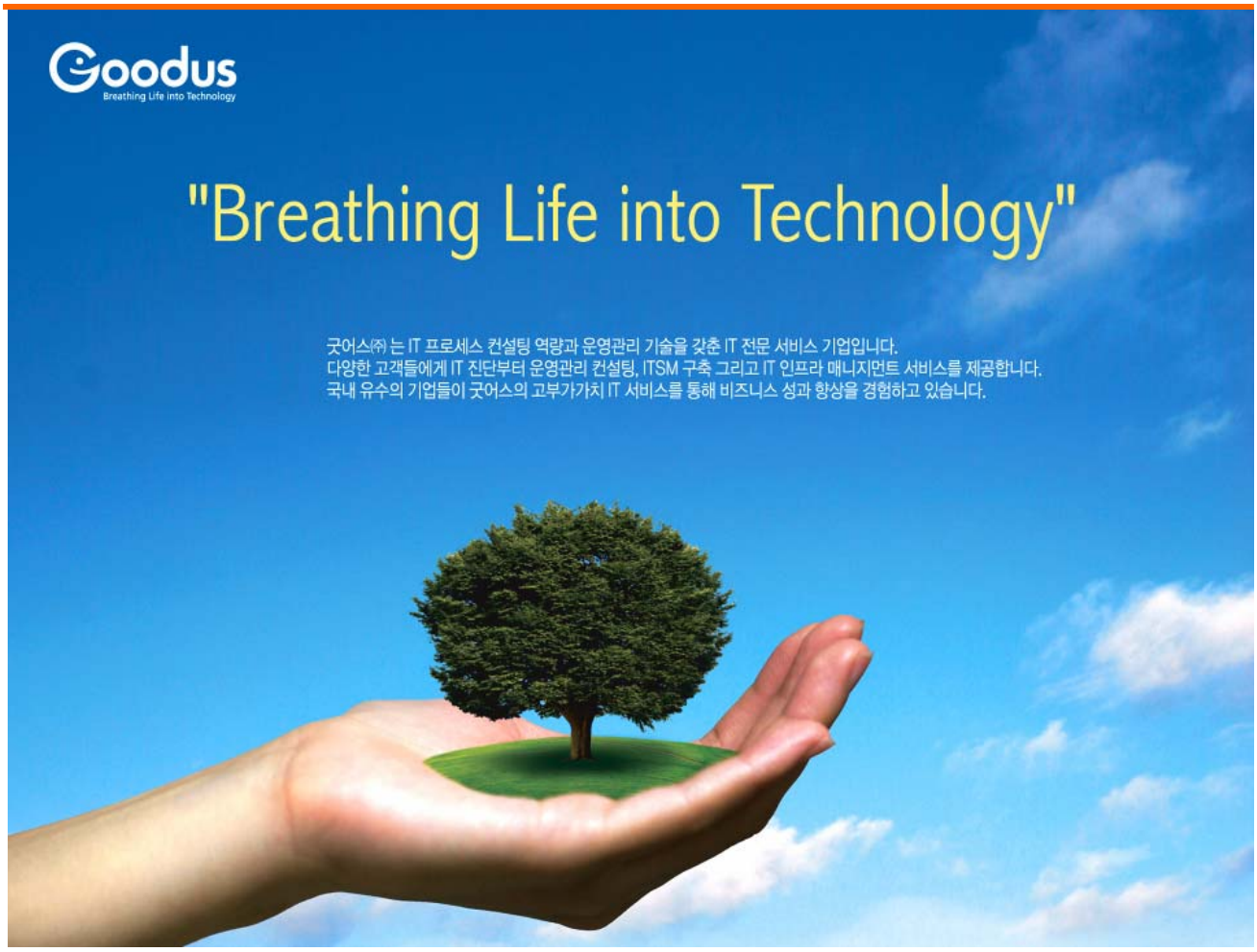
### 4.5.3. 사용자 관점

Proactive tuning service 는 계약 종료 시점 작업한 Tuning 산출물을 통한 기술전수 세미나를 진행 함으로서 고객사의 사용자가 실무에서 바로 적용 가능한 기술을 전수함과 동시에, 계약기간 종료 후에도 Proactive tuning service 를 유지 할 수 있는 방향을 제시 합니다.




### 4.5.4. 혁신적 관점




Goodus
Breathing Life into Technology
"Breathing Life into Technology"
굿어스㈜ 는 IT 프로세스 컨설팅 역량과 운영관리 기술을 갖춘 IT 전문 서비스 기업입니다.
다양한 고객들에게 IT 진단부터 운영관리 컨설팅, ITSM 구축 그리고 IT 인프라 매니지먼트 서비스를 제공합니다.
국내 유수의 기업들이 굿어스의 고부가가치 IT 서비스를 통해 비즈니스 성과 향상을 경험하고 있습니다.