ORACLE

오라클 Security - VPD

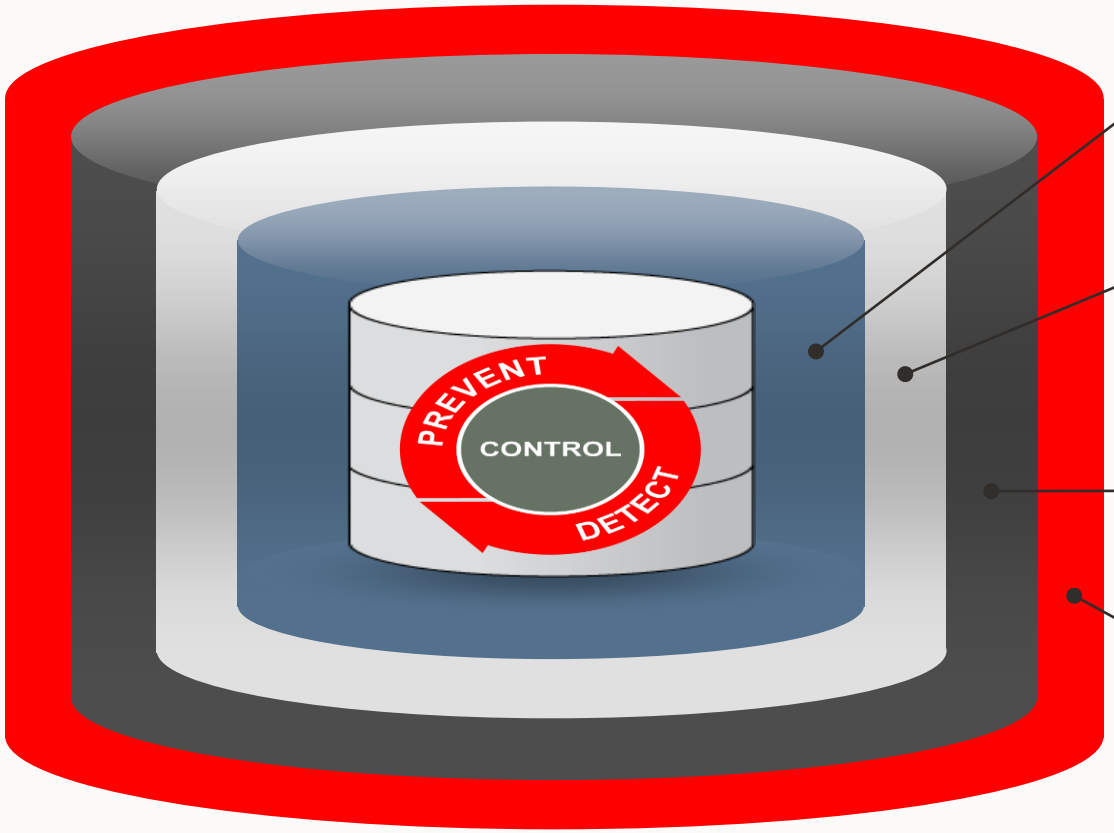
DB통합 환경에서 데이터 존 구성을 위한 VPD 이해 및 실습

고운용 상무
Master Principal Technical Architecture
Technical Solution Engineering, Oracle Korea

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

오라클 Security 사상
defense in depth – 데이터 근접 방호



Encryption and Masking
- Oracle Advanced Security / Key Vault
- Oracle Data Masking / redaction

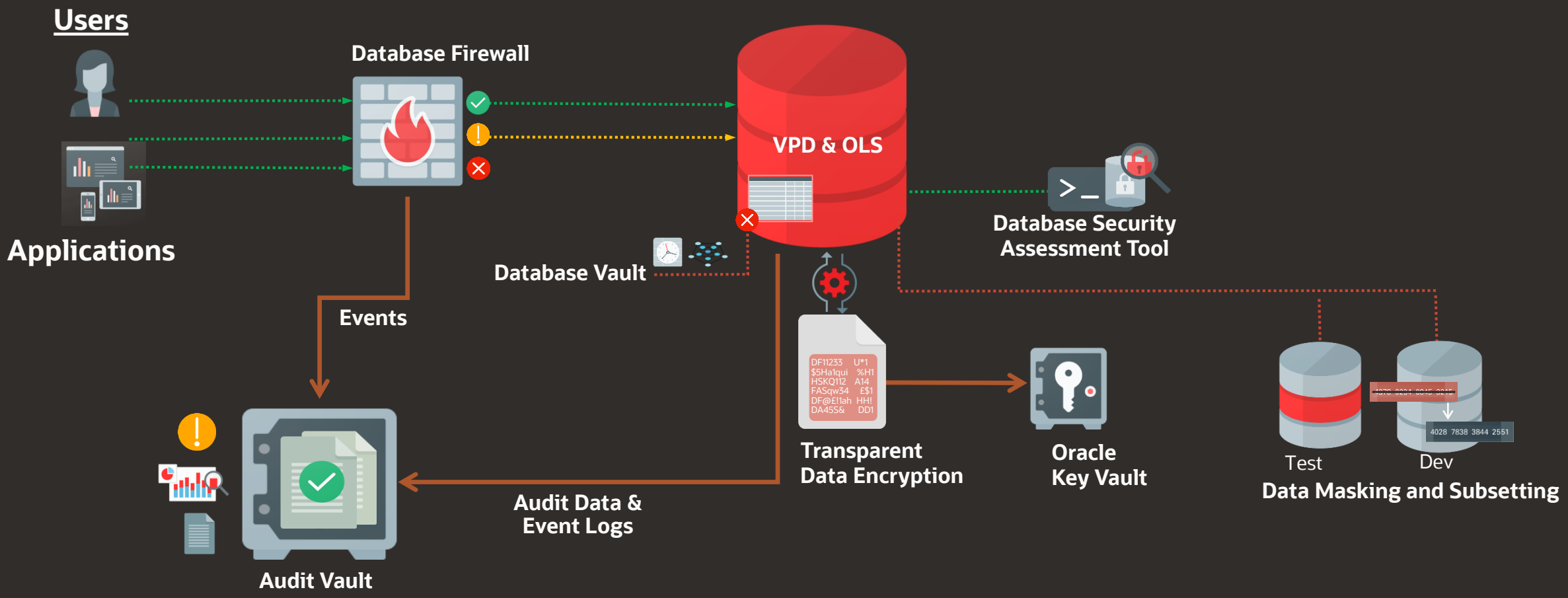
Access Control
- Oracle Database Vault / Key Vault
- Oracle Privileged Account Manager
- Oracle Label Security

Auditing and Tracking
- Oracle Audit Vault (AVDF)
- Oracle Configuration Management / Key Vault

Monitoring and Blocking
- Oracle Database Firewall (AVDF)

오라클 Security solution overview

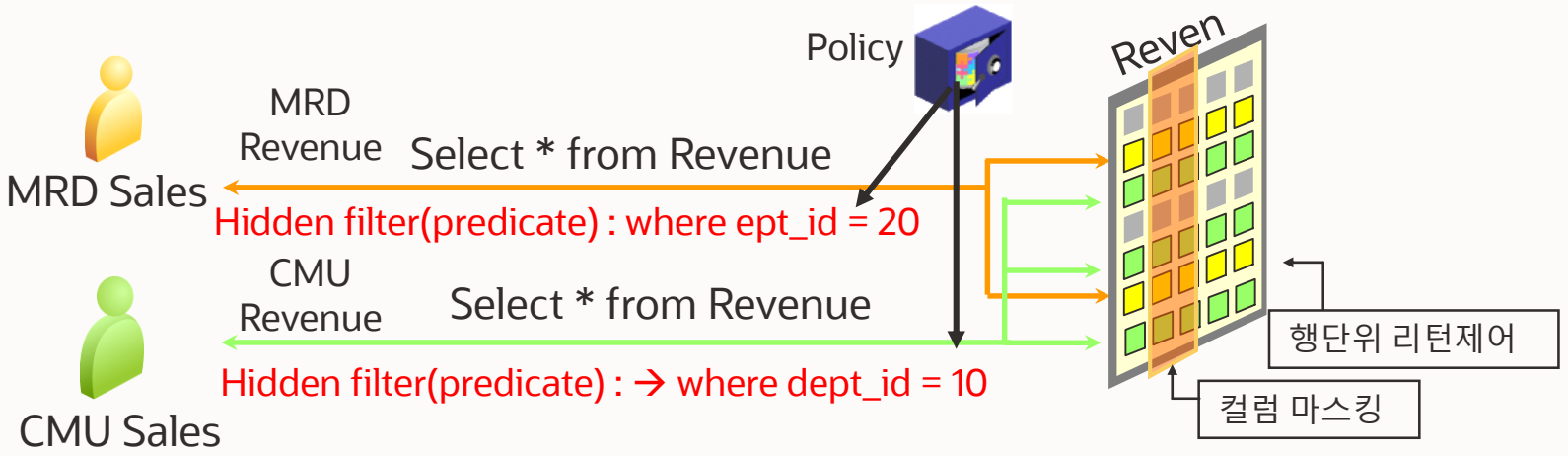
Maximum Security Architecture - on-premises



Virtual Private Database

VPD(Oracle Virtual Private Database) 요약

- Table, View 또는 Synonym에 동적 WHERE 절을 강제 적용
- Row 및 Column 레벨의 데이터 액세스를 제어하는 보안 기능
- 어플리케이션 통합환경에서 조직간의 데이터 존을 만들고 유지할 때 유용
- SELECT, INSERT, UPDATE, INDEX, 및 DELETE 구문 제어
- DDL(Truncate, Alter 등) 미 지원



VPD 강점

데이터 보안, 쉬운 구현, 유연한 운영

Security.

- Table, View 또는 synonym에 적용하여 잠재적인 심각한 보안 위협 문제 해결
- 어플리케이션에서 벗어난 다양한 **접근경로에 대한 데이터 접근 제어 가능**
- 통합 데이터 환경에서 데이터 Isolation(Private 데이터 존) 강제화 가능

Simplicity.

- **Once time**으로 적용

Flexibility.

- SQL 구문별(SELECT, INSERT, UPDATE, DELETE)로 정책 운영 가능
- 특정 조건에 맞는 Row 또는 컬럼 값만 접근허용 등에도 활용

VPD 구현 방법/절차

정책 설계 → 구현 → 운영

- 접근제어 대상 식별
  - 데이터(테이블, 컬럼)
  - DB 계정
  - 세션
- 접근제어 방법 식별
  - 제어(필터) 조건
  - 정책 타입 결정
    - Dynamic/static
    - context

- Policy Function 생성
- Policy 생성

- Policy Enable & Disable

VPD 세부 구성 기능
컴포넌트

1. Policy Function

- 동적 WHEWE 절을 강제 적용하도록 제어하는 함수
- [must] 동적 WHERE 절을 생성하기 위해 필요 (not procedure)

2. Policy

- Function을 붙이기 위한 정책
- DBMS_RLS 패키지를 사용하여 생성
  - 정책 관리 및 세분화된 접근제어 기능 내장
  - 특정 SQL 문 유형(SELECT, INSERT, UPDATE, DELETE, INDEX)에 대한 정책 시행 가능
  - SQL type 미지정시 SELECT, INSERT, UPDATE, DELETE 가 기본으로 작용(INDEX 제외)
- SYS 외의 사용자는 EXECUTE privilege 필요
- DBMS_RLS.ADD_POLICY 프로시저로 대상 객체에 적용

Oracle VPD Policy = VPD Function + Policy

VPD 세부 구성 기능
Policy Function 종류 및 기능

DYNAMIC policy

- 사용자 액세스 때마다 Policy Function 정책 평가
- VPD 디폴트 정책 임(Policy type 미선언시)
- 성능 최적화 보다는 정책 검증에 활용(처음 적용시 사용, 오라클 권고)

```
BEGIN
 DBMS_RLS.ADD_POLICY(
  object_schema   => 'hr',
  object_name     => 'employees',
  policy_name     => 'secure_update',
  policy_function => 'hide_fin',
  policy_type     => dbms_rls.DYNAMIC);
END;
/
```

STATIC & SHARED_STATIC policy

- 한번 정책 실행 후 SGA에 저장, 각 쿼리에 대해 매번 정책 평가 하지 않음(성능 최적화)
- DB 내 모든 사용자에 대해 동일 조건으로 적용할 때 사용
- DW와 같은 data host 환경에 유용
- Application context 의 “SYS_CONTEXT” 사용하여 조건 값 동적 변경 가능
- SHARED_STATIC, 여러 객체와 정책 공유 때 사용

```
BEGIN
 DBMS_RLS.ADD_POLICY(
  object_schema   => 'hr',
  object_name     => 'employees',
  policy_name     => 'secure_update',
  policy_function => 'hide_fin',
  policy_type     => dbms_rls.SHARED_STATIC);
END;
/
```

CONTEXT_SENSITIVE & SHARED_CONTEXT_SENSITIVE

- 2개 이상의 조건을 가지는 3tier 세션 풀링 어플리케이션
- 사용자 정보가 변경되지 않으면 정책 재평가(parsing) 하지 않음
- Namepsace와 Attribute 파라미터를 포함한 **Application context**로 제어
- Shared_context_Senditive 경우, UGA 통한 정책 공유

```
BEGIN
 DBMS_RLS.ADD_POLICY(
  object_schema   => 'hr',
  object_name     => 'employees',
  policy_name     => 'secure_update',
  policy_function => 'hide_fin',
  policy_type     => dbms_rls.CONTEXT_SENSITIVE,
  namespace       => 'empno_ctx',
  attribute       => 'emp_id');
END;
/
```

VPD 세부 구성기능

Policy Groups

- VPD 적용시 여러 세션들을 그룹화 하여 보안 정책을 적용할 수 있는 기능
- DBMS_RLS_ADD_POLICY_CONTEXT 프로시저로 Driving context 추가하게 됨
- 하나의 객체(Table, View, Synonym)에 여러 Driving Context 적용 가능하며, 각각은 개별적으로 처리

What is an Application Context?

- Oracle Database는 “Application Context”를 사용하여 데이터베이스 및 비데이터베이스 사용자에 대한 정보를 얻을 수 있음
- Application Context
  - Name-value pair의 집합, “namespace”라는 레이블이 있음
  - 정보 값은 메모리에 저장됨
- Application Context 정보를 이용하여 사용자가 애플리케이션을 통해 데이터에 액세스하는 것을 허용하거나 통제가 가능, 데이터베이스 및 비데이터베이스 사용자 모두 인증도 가능

형식
SYS_CONTEXT('Namespace','Parameter' [, Length])

Parameter로 제공되는 기본정보들(약 75종)

- CURRENT_SCHEMA
- DB_NAME
- HOST
- SERVER_HOST
- IP_ADDRESS
- GLOBAL_UID
- MODULE – application name
- OS_USER
- SESSION_USER
- PROXY_USER
- Etc…

사용자 지정 정보도 생성 가능

SQL Language Reference : SYS_CONTEXT 참조

URL: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/SYS_CONTEXT.html#GUID-B9934A5D-D97B-4E51-B01B-80C76A5BD086

VPD 샘플
“deptno = 30”을 제외한 Row들 리턴

```
CREATE OR REPLACE FUNCTION hide_sal_comm (
 v_schema IN VARCHAR2,
 v_objname IN VARCHAR2)

RETURN VARCHAR2 AS
con VARCHAR2 (200);

BEGIN
 con := 'deptno=30';
 RETURN (con);
END hide_sal_comm;
```

```
BEGIN
 DBMS_RLS.ADD_POLICY (
  object_schema     => 'scott',
  object_name       => 'emp',
  policy_name       => 'hide_sal_policy',
  policy_function   => 'hide_sal_comm',
  sec_relevant_cols => 'sal,comm');
END;
```

```
SELECT ENAME, d.dname, JOB, SAL, COMM
 FROM emp e, dept d
 WHERE d.deptno = e.deptno;
```

ENAME	DNAME	JOB	SAL	COMM
ALLEN	SALES	SALESMAN	1600	300
WARD	SALES	SALESMAN	1250	500
MARTIN	SALES	SALESMAN	1250	1400
BLAKE	SALES	MANAGER	2850	
TURNER	SALES	SALESMAN	1500	0
JAMES	SALES	CLERK	950	

6 rows selected.

VPD 샘플
Null값으로 컬럼값 마스킹된 상태로 Row들 리턴

```
CREATE OR REPLACE FUNCTION hide_sal_comm (
 v_schema IN VARCHAR2,
 v_objname IN VARCHAR2)

RETURN VARCHAR2 AS
con VARCHAR2 (200);

BEGIN
 con := 'deptno=30';
 RETURN (con);
END hide_sal_comm;
```

```
BEGIN
 DBMS_RLS.ADD_POLICY(
   object_schema       => 'scott',
   object_name         => 'emp',
   policy_name         => 'hide_sal_policy',
   policy_function     => 'hide_sal_comm',
   sec_relevant_cols   =>' sal,comm',
   sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
END;
```

```
SELECT ENAME, d.dname, job, sal, comm
 FROM emp e, dept d
 WHERE d.deptno = e.deptno;
```

Null값으로 마스킹

ENAME	DNAME	JOB	SAL	COMM
CLARK	ACCOUNTING	MANAGER		
KING	ACCOUNTING	PRESIDENT		
MILLER	ACCOUNTING	CLERK		
JONES	RESEARCH	MANAGER		
FORD	RESEARCH	ANALYST		
ADAMS	RESEARCH	CLERK		
SMITH	RESEARCH	CLERK		
SCOTT	RESEARCH	ANALYST		
WARD	SALES	SALESMAN	1250	500
TURNER	SALES	SALESMAN	1500	0
ALLEN	SALES	SALESMAN	1600	300
JAMES	SALES	CLERK	950	
BLAKE	SALES	MANAGER	2850	
MARTIN	SALES	SALESMAN	1250	1400

Oracle Virtual Private Database 실습

Application Context 기반 제어(basic)

실습1 시나리오 - Application Context(Basic) 이용한 접근제어 실습

실습 시나리오

- 각 소속 데이터관리자 계정은 Evaluate_t 테이블 Select, Insert, Update, Delete 가능
- But, 'DATA_ZONE' 컬럼 값과 evaluate_t.emp_sosok_cd 값이 같은 경우에 한해서 Row에 액세스 가능(소속별 데이터 존 구성)

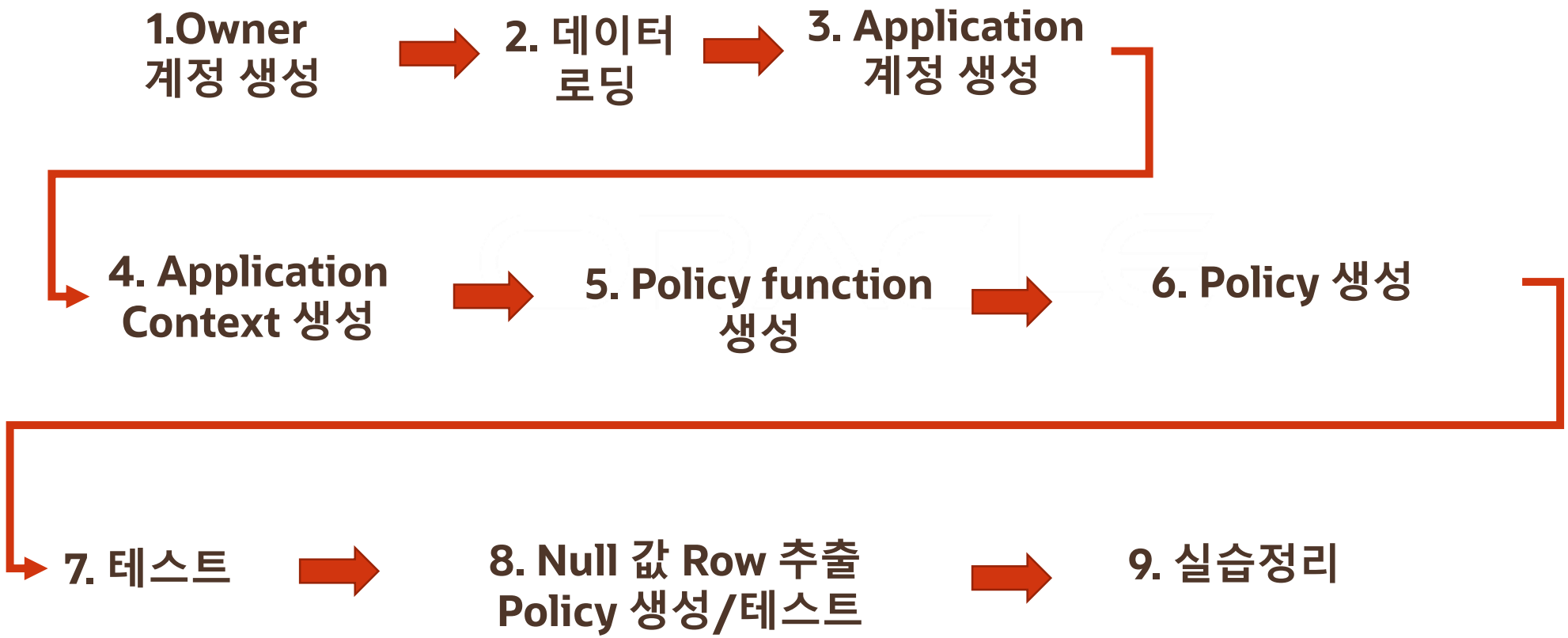
소속 구분 정보로 접근제어

사전 준비

- Linux 서버 정보 확인 및 터미널 접속 방법 확인
- 오라클 데이터 베이스 19c & 최신 패치 확인
- Instance 확인
- DB 서버 Oracle 계정 확인
- Instance의 SYS 정보 확인

이 실습은 오라클 멀티테넌트 인스턴스를 사용하여 실습이 진행됩니다.

진행 순서


1.Owner 계정 생성
2. 데이터 로딩
3. Application 계정 생성
4. Application Context 생성
5. Policy function 생성
6. Policy 생성
7. 테스트
8. Null 값 Row 추출 Policy 생성/테스트
9. 실습정리

1. Owner 계정 만들기

- VPD 실습을 위한 DB 계정은 "HANDSON"이며, password는 "HANDSON" 입니다.
- VPD 운영에 필요한 기본 권한은 CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE, CREATE TRIGGER, ADMINISTER DATABASE TRIGGER, EXEMPT ACCERSS POLICY과 EXEXUTE on dbms_session, EXECUTE on DBMS_RLS입니다.
- 아래 스크립트를 이용하여 DB 계정을 만들고 및 권한을 부여합니다.

DB 서버 접속 : oracle
Linux> . oraenv [instance명]

```
sqlplus sys/Welcome1@pdbxx as sysdba

-- alter session set container=pdbxx     /* pluggable 경우
-- show con_name
create user handson identified by handson default tablespace users container=current
quota unlimited on users;
grant CONNECT, RESOURCE to handson container=current;

set line 150
col username format a20
col account_status format a10

select username, account_status, created from dba_users where username = 'HANDSON';
```

2. 데이터 로딩

2.1 실습 테이블 생성

- VPD 실습을 위한 DB 계정은 "HANDSON"이며, password는 "HANDSON" 입니다.
- VPD 운영에 필요한 기본 권한은 CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE, CREATE TRIGGER, ADMINISTER DATABASE TRIGGER, EXEMPT ACCERSS POLICY과 EXEXUTE on dbms_session, EXECUTE on DBMS_RLS입니다.
- 아래 스크립트를 이용하여 DB 계정을 만들고 및 권한을 부여합니다.

```
SQL> conn handson/handson@pdbxx

create table evaluate_t (empno varchar2(12)
not null,
fst_ev_grade varchar2(2),
snd_ev_grade varchar2(2),
fin_ev_grade varchar2(2),
emp_sosok_cd  varchar2(4) not null,
emp_sb_cd varchar2(6) not null);

create table account_t (
acct_id varchar2(20) not null,
data_zone varchar2(20) not null,
sosok varchar2(20),
sosok_br varchar(20),
s_role varchar2(20));

col table_name format a30
select table_name, status from user_tables where table_name in ('EVALUATE_T','ACCOUNT_T');

TABLE_NAME                     STATUS
------------------------------ ------------------------
ACCOUNT_T                      VALID
EVALUATE_T                     VALID
```

2. 데이터 로딩

2.2 데이터 로딩

- VPD 실습을 위해 별도 제공되는 쉘스크립트와 가상데이터를 사용하여 위에서 만든 테이블에 데이터를 입력합니다. 데이터 로딩 툴은 SQL*LOADER를 사용합니다.
- 별도 제공되는 쉘스크립트와 가상데이터 목록은 다음과 같습니다.

  - account_dataset4vpd.dat -- account_t 테이블을 위한 데이터 셋
  - Evaluate_dataset.dat -- evaluate_t 테이블을 위한 데이터 셋
  - load_acc.ctl -- account_t 테이블의 데이터 로딩 control 파일
  - load_acc.sh -- account_t 테이블의 데이터 로딩 쉘
  - load_evl.ctl --evaluate_t 테이블의 데이터 로딩 control 파일
  - load_evl.sh -- evaluate_t 테이블의 데이터 로딩 쉘

2. 데이터 로딩

2.2 데이터 확인

```
sqlplus handson/handson@pdbxx
set pagesize 100
set line 200
col acct_id format a20
col data_zone format a20

select acct_id, data_zone from account_t;

ACCT_ID              DATA_ZONE
-------------------- ----------
ADMINAR              AR
ADMINARMC            ARMC
ADMINARAD            ARMIL
ADMINNV              NV
ADMINAF              AF
```

```
select emp_sosok_cd, count(*) from evaluate_t group by emp_sosok_cd;
EMP_MIL_CD    COUNT(*)
------------ ----------
AR                   10
NV                   10
AF                   10
```

```
select emp_sb_cd, count(*) from evaluate_t where emp_sosok_cd = 'AR' and
emp_sb_cd in ('MIL','MC') group by emp_sb_cd;

EMP_SB_CD          COUNT(*)
------------------ ----------
MC                          2
MIL                         8
```

```
select * from evaluate_t;
EMPNO       FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
----------- ------ ------ ------ ------------ ------------------
21-10011    4      4             AR           MIL
21-10102    1      2      2      AR           MIL
21-10203    2      2      2      AR           MIL
21-10044    3      3      2      AR           MC
21-15005    3      3      3      AR           MIL
21-10106    1      5             AR           MC
21-10307    2      1      1      AR           MIL
21-10038    2      2      2      AR           MIL
21-10009    1      2      2      AR           MIL
~
~
21-31102    2      2      2      AF           MIL
21-33203    2      2      2      AF           MIL
21-30004    2      3             AF           MC
21-30905    2      1      1      AF           MIL
21-30806    1      2             AF           MIL
21-30047    2      2      2      AF           MC
21-30078    3      2      2      AF           MIL
21-30309    3      3      3      AF           MIL
21-30010    2      2      2      AF           MIL
```

3. Application 계정 만들기

3.1 Application 계정 만들기

VPD 실습에 사용되는 APPUSER, ADMINAR, ADMINARMC, ADMINARNIL, ADMINNV, ADMINAF 등 6개의 DB 계정을 사용합니다. 아래 스크립트를 이용하여 DB 계정을 생성합니다. APPUSER는 Application 계정이고 ADMINxxx는 소속별 데이터 관리자입니다,

```
conn sys/Welcome1@pdbxx as sysdba

show user;

CREATE USER appuser IDENTIFIED BY welcome1 DEFAULT TABLESPACE users CONTAINER=CURRENT;

GRANT connect, resource to appuser CONTAINER=CURRENT;

GRANT CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE, CREATE TRIGGER, ADMINISTER DATABASE TRIGGER TO
appuser CONTAINER=CURRENT;

GRANT EXECUTE ON DBMS_SESSION TO appuser CONTAINER=CURRENT;

GRANT EXECUTE ON DBMS_RLS TO appuser CONTAINER=CURRENT;
```

3. Application 계정 만들기

3.2 소속별 데이터 관리자 계정 만들기

```
create user adminar identified by welcome1 default tablespace users container=current;
grant connect to adminar container=current;
create user adminnv identified by welcome1 default tablespace users container=current;
grant connect to adminnv container=current;
create user adminaf identified by welcome1 default tablespace users container=current;
grant connect to adminaf container=current;
create user adminarmil identified by welcome1 default tablespace users container=current;
grant connect to adminarmil container=current;
create user adminarmc identified by welcome1 default tablespace users container=current;
grant connect to adminarmc container=current;
```

3. Application 계정 만들기

3.3 액세스 권한 부여
conn handson/handson@pdbxx

grant select, insert, update, delete on account_t to appuser, adminar, adminnv, adminaf, adminarmil, adminarmc container=current;

grant select, insert, update, delete on evaluate_t to appuser, adminar, adminnv, adminaf, adminarmil, adminarmc container=current;

3. Application 계정 만들기

3.3 데이터 확인

```
conn appuser/welcome1@pdbxx

column empno format a12
select * from handson.evaluate_t ;

EMPNO       FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
----------- ------ ------ ------ ------------ ------------------
21-10011    4      4             AR           MIL
21-10102    1      2      2      AR           MIL
21-10203    2      2      2      AR           MIL
21-10044    3      3      2      AR           MC
21-15005    3      3      3      AR           MIL
21-10106    1      5             AR           MC
21-10307    2      1      1      AR           MIL
~
~
21-30806    1      2             AF           MIL
21-30047    2      2      2      AF           MC
21-30078    3      2      2      AF           MIL
21-30309    3      3      3      AF           MIL
21-30010    2      2      2      AF           MIL

30 행이 선택되었습니다.
```

```
conn adminarmil/welcome1@pdbxx

column empno format a12
select * from handson.evaluate_t ;

EMPNO       FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
----------- ------ ------ ------ ------------ ------------------
21-10011    4      4             AR           MIL
21-10102    1      2      2      AR           MIL
21-10203    2      2      2      AR           MIL
21-10044    3      3      2      AR           MC
21-15005    3      3      3      AR           MIL
21-10106    1      5             AR           MC
21-10307    2      1      1      AR           MIL
21-10038    2      2      2      AR           MIL
~
~
21-30078    3      2      2      AF           MIL
21-30309    3      3      3      AF           MIL
21-30010    2      2      2      AF           MIL

30 행이 선택되었습니다.
```

똑 같은 방법으로 adminaf, adminnv 등의 다른 계정으로 로그인 하여 evaluate_t 테이블의 데이터가 조회되는지 확인 합니다.

3. Application 계정 만들기

```
conn appuser/welcome1@pdbxx

column empno format a12

select acct_id, data_zone from handson.account_t;

ACCT_ID              DATA_ZONE
-------------------- ----------
ADMINAR              AR
ADMINARMC            ARMC
ADMINARAD            ARMIL
ADMINNV              NV
ADMINAF              AF
;
```

4. Application context 생성

4.1 Application Context 생성

- Oracle Database는 “Application Context”를 사용하여 데이터베이스 및 비데이터베이스 사용자에 대한 정보를 얻을 수 있습니다. 이 정보를 이용하여 Application 및 SQL*PLUS와 같은 DB 액세스 툴을 통해 데이터에 액세스 하는 것을 허용하거나 통제가 가능합니다.
- 아래와 같이 Application context 를 위한 작업을 수행합니다.
- Application 계정이 DB에 로그인 할 때 Application Context에 사용할 사용자 세션 정보를 강제 적용하도록 합니다. 이 정보는 Application 계정이 수정을 못하도록 DB 커널에서 강제됩니다.

```
conn appuser/welcome1@pdbxx

col dbuser format a20

select SYS_CONTEXT('USERENV', 'SESSION_USER') dbuser
from dual;

DBUSER
--------------------
APPUSER
```

```
CREATE OR REPLACE CONTEXT datazone_ctx USING datazone_ctx_pkg;

CREATE OR REPLACE PACKAGE datazone_ctx_pkg IS
 PROCEDURE set_datazone;
END;
/
CREATE OR REPLACE PACKAGE BODY datazone_ctx_pkg IS
 PROCEDURE set_datazone
 AS
  datazone varchar2(20);
 BEGIN
   SELECT data_zone INTO datazone FROM handson.account_t
     WHERE acct_id = SYS_CONTEXT('USERENV', 'SESSION_USER');
     DBMS_SESSION.SET_CONTEXT('datazone_ctx', 'data_zone', datazone);
 EXCEPTION
  WHEN NO_DATA_FOUND THEN NULL;
 END set_datazone;
END;
/
```

4. Application context 생성

4.2 세션 정보 트리거 생성

Application context 정보를 User가 DB 로그인 때 강제로 세션 정보를 할당하기 위해 트리거 방식을 사용합니다. 트리거를 아래와 같이 생성합니다.

```
conn appuser/welcome1@pdbxx

CREATE or REPLACE TRIGGER set_datazone_ctx_trig AFTER LOGON ON DATABASE
 BEGIN
    appuser.datazone_ctx_pkg.set_datazone;
  EXCEPTION
  WHEN OTHERS THEN
   RAISE_APPLICATION_ERROR(
    -20000, 'Trigger handson.datazone_ctx_pkg.set_datazone violation. Login denied.');
 END;
/
```

4.3 Application context 정보 확인

```
conn adminar/welcome1@pdbxx

SELECT SYS_CONTEXT('datazone_ctx', 'data_zone') DBzone FROM DUAL;

DBZONE
--------------------------------------------------------------------------------
AR


conn adminarmil/welcome1@pdbxx

SELECT SYS_CONTEXT('datazone_ctx', 'data_zone') DBzone FROM DUAL;

DBZONE
--------------------------------------------------------------------------------
ARMIL
```

5. Policy function 생성

```
conn appuser/welcome1@pdbxx

CREATE OR REPLACE FUNCTION vpd_get_datazone(
  schema_p   IN VARCHAR2,
  table_p    IN VARCHAR2)
 RETURN VARCHAR2
 AS
  users_pred VARCHAR2 (400);
 BEGIN
                users_pred := 'emp_sosok_cd = SYS_CONTEXT(''datazone_ctx'', ''data_zone'')';
 RETURN users_pred;
END;
/
```

6. VPD Policy 생성

```
conn appuser/welcome1@pdbxx

Begin
dbms_rls.add_policy(
object_schema => 'HANDSON',
object_name => 'EVALUATE_T',
policy_name =>'EVAL_DATAZONE_POLICY',
Function_schema => 'APPUSER',
policy_function =>'VPD_GET_DATAZONE',
Statement_types => 'SELECT,UPDATE,INSERT,DELETE',
policy_type => DBMS_RLS.CONTEXT_SENSITIVE,
namespace => 'DATAZONE_CTX',
attribute=>'DATA_ZONE',
update_check => TRUE,
enable => TRUE);
end;
/
```

7. Policy 테스트

7.1 자신의 데이터 존 영역의 데이터 조회

```
conn adminar / welcome1@pdbxx
SELECT SYS_CONTEXT('datazone_ctx', 'data_zone') DATAZONE FROM DUAL;

DATAZONE
--------------------------------------------------------------------------------
AR

select * from handson.evaluate_t;

EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ----------------
21-10011                        4      4             AR           MIL
21-10102                        1      2      2      AR           MIL
21-10203                        2      2      2      AR           MIL
21-10044                        3      3      2      AR           MC
21-15005                        3      3      3      AR           MIL
21-10106                        1      5             AR           MC
21-10307                        2      1      1      AR           MIL
21-10038                        2      2      2      AR           MIL
21-10009                        1      2      2      AR           MIL
21-11010                        2      3      3      AR           MIL

10 행이 선택되었습니다.
```

```
conn adminaf/welcome1@pdbxx

select * from handson.evaluate_t ;

EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ----------------
21-30001                        1      2      2      AF           MIL
21-31102                        2      2      2      AF           MIL
21-33203                        2      2      2      AF           MIL
21-30004                        2      3             AF           MC
21-30905                        2      1      1      AF           MIL
21-30806                        1      2             AF           MIL
21-30047                        2      2      2      AF           MC
21-30078                        3      2      2      AF           MIL
21-30309                        3      3      3      AF           MIL
21-30010                        2      2      2      AF           MIL
```

7. Policy 테스트

7.2 타 데이터 존의 데이터 조회

```
conn adminarmil/welcome1@pdbxx

select * from handson.evaluate_t;

선택된 레코드가 없습니다.

column datazone format a20

SELECT SYS_CONTEXT('datazone_ctx', 'data_zone') DATAZONE FROM DUAL;

DATAZONE
--------------------
ARMIL
```

8. Null값으로 마스킹된 Row들 리턴

8.1 Policy 재생성

```
conn appuser/welcome1@pdbxx

exec DBMS_RLS.DROP_POLICY('handson', 'evaluate_t', 'eval_datazone_policy');

Begin
dbms_rls.add_policy(
object_schema => 'HANDSON',
object_name => 'EVALUATE_T',
policy_name =>'EVAL_MASKING_POLICY',
Function_schema => 'APPUSER',
policy_function =>'VPD_GET_DATAZONE',
policy_type => DBMS_RLS.CONTEXT_SENSITIVE,
namespace => 'DATAZONE_CTX',
attribute=>'DATA_ZONE',
update_check => TRUE,
enable =>TRUE,
sec_relevant_cols => 'FST_EV_GRADE,SND_EV_GRADE,FIN_EV_GRADE',
sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS);
end;
/
```

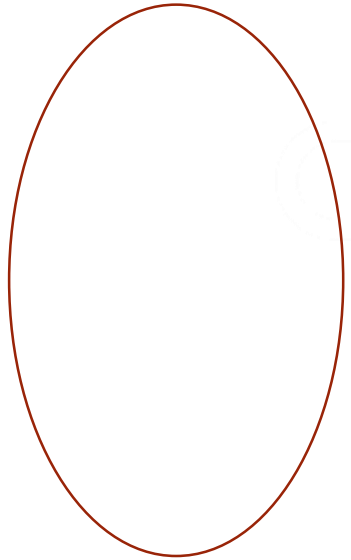
8. Null값으로 마스킹된 Row들 리턴

8.2 NULL 값 마스킹 확인

```
select * from handson.evaluate_t;

EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ----------------
21-10011                                             AR           MIL
21-10102                                             AR           MIL
21-10203                                             AR           MIL
21-10044                                             AR           MC
21-15005                                             AR           MIL
21-10106                                             AR           MC
21-10307                                             AR           MIL
21-10038                                             AR           MIL
21-10009                                             AR           MIL
21-11010                                             AR           MIL
21-20301                                             NV           MIL
21-20002                                             NV           MIL
21-20343                                             NV           MIL
~
~
21-33203                                             AF           MIL
21-30004                                             AF           MC
21-30905                                             AF           MIL
21-30806                                             AF           MIL
21-30047                                             AF           MC
21-30078                                             AF           MIL
21-30309                                             AF           MIL
21-30010                                             AF           MIL

30 행이 선택되었습니다.
```



Null 값으로 마스킹 된 결과

9. 실습 정리

```
conn appuser/welcome1@pdbxx

drop package datazone_ctx_pkg;
drop function vpd_get_datazone;
drop trigger set_datazone_ctx_trig;
exec DBMS_RLS.DROP_POLICY('handson', 'evaluate_t', 'eval_datazone_policy');
exec DBMS_RLS.DROP_POLICY('handson', 'evaluate_t', 'EVAL_MASKING_POLICY');

conn sys/Welcome1@pdbxx as sysdba
drop context datazone_ctx;

drop context datazone_ctx;
drop context datazone_drv_ctx;
drop user handson cascade;
drop user ADMINAR cascade;
drop user ADMINARMC cascade;
drop user ADMINARMIL cascade;
drop user ADMINNV cascade;
drop user ADMINAF cascade;
```

Oracle Virtual Private Database 실습

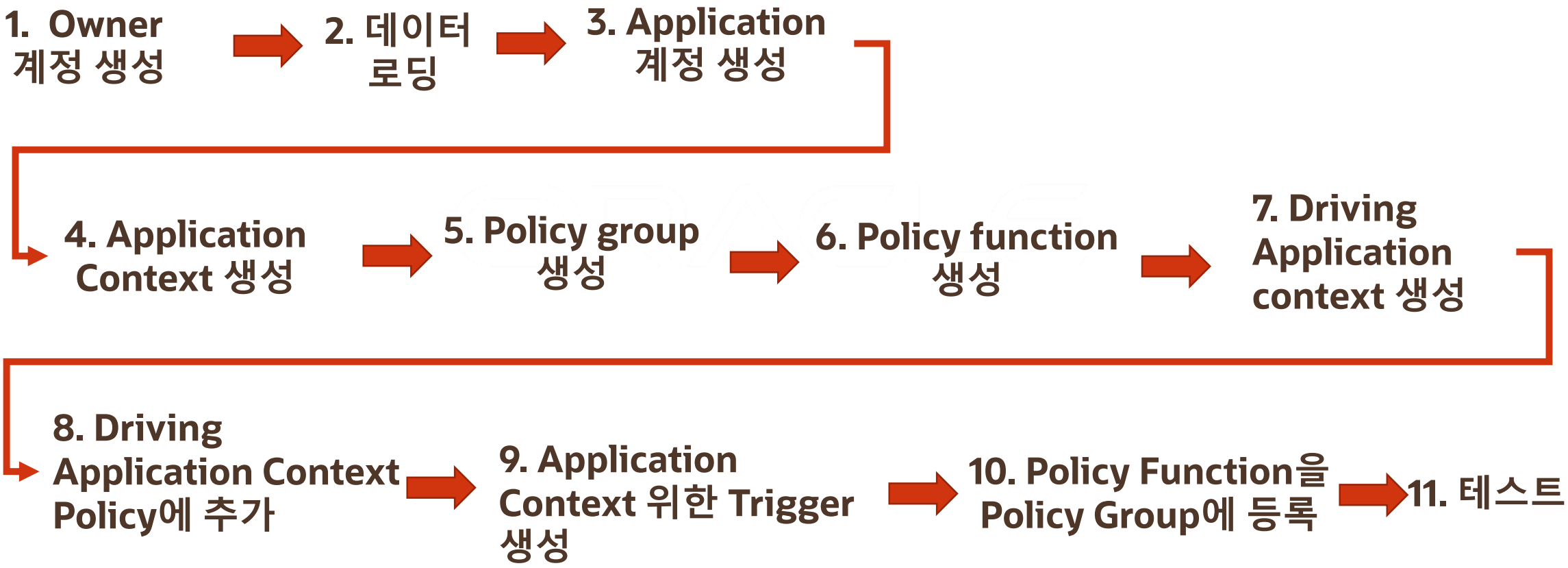
Application Context 기반 제어(Policy group)

실습 시나리오

- 각 데이터관리자는 data_zone 에 명시된 조건에 따라 evaluate_t 테이블의 row에만 접근 허용

- ADMINAR*** 데이터 관리자를 위한 소속(emp_sosok_cd) 과 신분(emp_sb_cd) 기준의 데이터 존 구성

➡ 신분, 소속에 따른 접근제어


1. Owner 계정 생성
2. 데이터 로딩
3. Application 계정 생성
4. Application Context 생성
5. Policy group 생성
6. Policy function 생성
7. Driving Application context 생성
8. Driving Application Context Policy에 추가
9. Application Context 위한 Trigger 생성
10. Policy Function을 Policy Group에 등록
11. 테스트

앞장의 실습 준비 자료 참조

4. Application Context 생성

```
conn appuser/welcome1@pdbxx

CREATE OR REPLACE CONTEXT datazone_ctx USING datazone_ctx_pkg;

CREATE OR REPLACE PACKAGE datazone_ctx_pkg IS
  PROCEDURE set_datazone;
 END;
/
CREATE OR REPLACE PACKAGE BODY datazone_ctx_pkg IS
  PROCEDURE set_datazone
  AS
    datazone varchar2(20);
  BEGIN
     SELECT data_zone INTO datazone FROM handson.account_t
       WHERE acct_id = SYS_CONTEXT('USERENV', 'SESSION_USER');
       DBMS_SESSION.SET_CONTEXT('datazone_ctx', 'data_zone', datazone);
  EXCEPTION
   WHEN NO_DATA_FOUND THEN NULL;
  END set_datazone;
END;
/
```

5. Policy group 생성

```
conn appuser/welcome1@pdbxx

BEGIN
 DBMS_RLS.CREATE_POLICY_GROUP(
 object_schema   => 'handson',
 object_name     => 'evaluate_t',
 policy_group    => 'sosok_group');
END;
/

BEGIN
 DBMS_RLS.CREATE_POLICY_GROUP(
 object_schema   => 'handson',
 object_name     => 'evaluate_t',
 policy_group    => 'sb_group');
END;
/
```

6. Policy function 생성

```
conn appuser/welcome1@pdbxx

CREATE OR REPLACE FUNCTION vpd_function_sosok_group
 (schema in varchar2, tab in varchar2) return varchar2
 as predicate  varchar2(1000) default NULL ;
  BEGIN
   IF LOWER(SYS_CONTEXT('datazone_drv_ctx','policy_group')) = 'sosok_group'
      THEN predicate := 'emp_sosok_cd = SYS_CONTEXT(''datazone_ctx'',''data_zone'')';
   ELSE Null;
  END IF;
  RETURN predicate;
END;
/


CREATE OR REPLACE FUNCTION vpd_function_sb_group
 (schema in varchar2, tab in varchar2) return varchar2 as
  predicate  varchar2(2000) default NULL;
  BEGIN
   IF LOWER(SYS_CONTEXT('datazone_drv_ctx','policy_group')) = 'sb_group'
    THEN predicate := 'emp_sb_cd = SUBSTR(SYS_CONTEXT(''datazone_ctx'',''data_zone''),3,3) and
    emp_sosok_cd = SUBSTR(SYS_CONTEXT(''datazone_ctx'',''data_zone''),1,2)';
   ELSE Null;
  END IF;
  RETURN predicate;
END;
/
```

7. Driving Application context 생성

```
conn appuser/welcome1@pdbxx

CREATE OR REPLACE CONTEXT datazone_drv_ctx USING datazone_drv_ctx_pkg;

CREATE OR REPLACE PACKAGE datazone_drv_ctx_pkg IS
  PROCEDURE set_drv_context (policy_group varchar2 default NULL);
 END;
/
CREATE OR REPLACE PACKAGE BODY datazone_drv_ctx_pkg IS
  PROCEDURE set_drv_context (policy_group varchar2 default NULL) IS
  BEGIN
  CASE LOWER(SYS_CONTEXT('datazone_ctx', 'data_zone'))
    WHEN 'ar' THEN
       DBMS_SESSION.SET_CONTEXT('datazone_drv_ctx','policy_group','SOSOK_GROUP');
    WHEN 'nv' THEN
       DBMS_SESSION.SET_CONTEXT('datazone_drv_ctx','policy_group','SOSOK_GROUP');
    WHEN 'af' THEN
       DBMS_SESSION.SET_CONTEXT('datazone_drv_ctx','policy_group','SOSOK_GROUP');
    WHEN 'armil' THEN
       DBMS_SESSION.SET_CONTEXT('datazone_drv_ctx','policy_group','SB_GROUP');
    WHEN 'armc' THEN
       DBMS_SESSION.SET_CONTEXT('datazone_drv_ctx','policy_group','SB_GROUP');
  END CASE;
  END set_drv_context;
END;
/
```

8. Driving Application context를 context 에 추가

```
conn appuser/welcome1@pdbxx

BEGIN
 DBMS_RLS.ADD_POLICY_CONTEXT(
 object_schema  =>'handson',
 object_name    =>'evaluate_t',
 namespace      =>'datazone_drv_ctx',
 attribute      =>'policy_group');
END;
/
```

9.Driving Application context를 위한 Trigger 생성

9.1 세션정보 트리거 생성

conn appuser/welcome1@pdbxx

```
CREATE or REPLACE TRIGGER set_datazone_ctx_trig AFTER LOGON ON DATABASE
 BEGIN
    appuser.datazone_ctx_pkg.set_datazone;
   EXCEPTION
   WHEN OTHERS THEN
   RAISE_APPLICATION_ERROR(
    -20000, 'Trigger handson.datazone_ctx_pkg.set_datazone violation. Login denied.');
 END;
 /

 CREATE or REPLACE TRIGGER set_datazone_drv_ctx_trig AFTER LOGON ON DATABASE
 BEGIN
    appuser.datazone_drv_ctx_pkg.set_drv_context;
  EXCEPTION
  WHEN OTHERS THEN
   RAISE_APPLICATION_ERROR(
    -20000, 'Trigger handson. datazone_drv_ctx_pkg.set_drv_context violation. Login denied.');
 END;
/
```

9.Driving Application context를 위한 Trigger 생성

9.2 세션정보 확인

```
conn adminarmil/welcome1@pdbxx

-- exec handson.datazone_drv_ctx_pkg.set_drv_context;
col datazone format a20
select sys_context('datazone_ctx', 'data_zone') DATAZONE from dual;

DATAZONE
--------------------
ARMIL


col DriveGroup format a20
select sys_context('datazone_drv_ctx','policy_group') DriveGroup from dual;

DRIVEGROUP
--------------------
SB_GROUP
```

10. Policy function을 Policy group에 등록

```
conn appuser/welcome1@pdbxx

BEGIN
 DBMS_RLS.ADD_GROUPED_POLICY(
 object_schema        => 'handson',
 object_name          => 'evaluate_t',
 policy_group         => 'sosok_group',
 policy_name          => 'filter_sosok_policy',
 function_schema      => 'appuser',
 policy_function      => 'vpd_function_sosok_group',
 statement_types      => 'select',
 policy_type          => DBMS_RLS.CONTEXT_SENSITIVE,
 namespace            => 'datazone_drv_ctx',
 attribute            => 'policy_group');
END;
/
```

```
conn appuser/welcome1@pdbxx

BEGIN
 DBMS_RLS.ADD_GROUPED_POLICY(
 object_schema        => 'handson',
 object_name          => 'evaluate_t',
 policy_group         => 'sb_group',
 policy_name          => 'filter_sb_policy',
 function_schema      => 'appuser',
 policy_function      => 'vpd_function_sb_group',
 statement_types      => 'select',
 policy_type          => DBMS_RLS.CONTEXT_SENSITIVE,
 namespace            => 'datazone_drv_ctx',
 attribute            => 'policy_group' );
END;
/
```

11. Policy group 이용한 Policy에 대한 테스트

```
conn adminar/welcome1@pdbxx
set line 200
select * from handson.evaluate_t ;
```

```
EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ------------------
21-10011                        4      4             AR           MIL
21-10102                        1      2      2      AR           MIL
21-10203                        2      2      2      AR           MIL
21-10044                        3      3      2      AR           MC
21-15005                        3      3      3      AR           MIL
21-10106                        1      5             AR           MC
21-10307                        2      1      1      AR           MIL
21-10038                        2      2      2      AR           MIL
21-10009                        1      2      2      AR           MIL
21-11010                        2      3      3      AR           MIL

10 행이 선택되었습니다.
```

```
conn adminaf/welcome1@pdbxx
set line 200

select * from handson.evaluate_t ;
```

```
EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ------------------
21-30001                        1      2      2      AF           MIL
21-31102                        2      2      2      AF           MIL
21-33203                        2      2      2      AF           MIL
21-30004                        2      3             AF           MC
21-30905                        2      1      1      AF           MIL
21-30806                        1      2             AF           MIL
21-30047                        2      2      2      AF           MC
21-30078                        3      2      2      AF           MIL
21-30309                        3      3      3      AF           MIL
21-30010                        2      2      2      AF           MIL

10 행이 선택되었습니다.
```

11. Policy group 이용한 Policy에 대한 테스트

```
conn adminarmil/welcome1@pdbxx
set line 200
select * from handson.evaluate_t ;
```

```
EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ----------
21-10011                        4      4             AR           MIL
21-10102                        1      2      2      AR           MIL
21-10203                        2      2      2      AR           MIL
21-15005                        3      3      3      AR           MIL
21-10307                        2      1      1      AR           MIL
21-10038                        2      2      2      AR           MIL
21-10009                        1      2      2      AR           MIL
21-11010                        2      3      3      AR           MIL

8 행이 선택되었습니다.
```

```
conn adminarmc/welcome1@pdbxx
set line 200

select * from handson.evaluate_t ;
EMPNO                          FST_EV SND_EV FIN_EV EMP_SOSOK_CD EMP_SB_CD
------------------------------ ------ ------ ------ ------------ ----------
21-10044                        3      3      2      AR           MC
21-10106                        1      5             AR           MC
```

실습 자료 정리

```
conn appuser/welcome1@pdbxx

drop package DATAZONE_DRV_CTX_PKG;
drop function VPD_FUNCTION_SOSOK_GROUP;
drop function VPD_FUNCTION_SB_GROUP;
drop package DATAZONE_CTX_PKG;

drop trigger SET_DATAZONE_CTX_TRIG;
drop trigger SET_DATAZONE_DRV_CTX_TRIG;

exec
DBMS_RLS.DISABLE_GROUPED_POLICY('HANDSON','EVALUATE_T','SOSOK_GROUP','
FILTER_SOSOK_POLICY');
exec
DBMS_RLS.DISABLE_GROUPED_POLICY('HANDSON','EVALUATE_T','SB_GROUP','FILT
ER_SB_POLICY');

exec DBMS_RLS.DROP_GROUPED_POLICY('HANDSON', 'EVALUATE_T',
'SOSOK_GROUP','FILTER_SOSOK_POLICY') ;
exec DBMS_RLS.DROP_GROUPED_POLICY('HANDSON', 'EVALUATE_T',
'SB_GROUP','FILTER_SB_POLICY') ;

exec
DBMS_RLS.DELETE_POLICY_GROUP('HANDSON','EVALUATE_T','SOSOK_GROUP');
exec DBMS_RLS.DELETE_POLICY_GROUP('HANDSON','EVALUATE_T','SB_GROUP');

exec
DBMS_RLS.DROP_POLICY_CONTEXT('handson','evaluate_t','datazone_drv_ctx','policy
_group');
```

```
conn sys/Welcome1@pdbxx as sysdba

drop context datazone_ctx;
drop context datazone_drv_ctx;
drop user handson cascade;
Drop user appuser cascade;
drop user ADMINAR cascade;
drop user ADMINARMC cascade;
drop user ADMINARMIL cascade;
drop user ADMINNV cascade;
drop user ADMINAF cascade;
```

수고하셨습니다.

Woonyong.ko@oracle.com