Oracle 보안

TDE

Author	고요한
Creation Date	2011.12.05
Last Updated	2013-04-12
Version	V2.0
Copyright(C) 2004 Goodus Inc. All Rights Reserved	

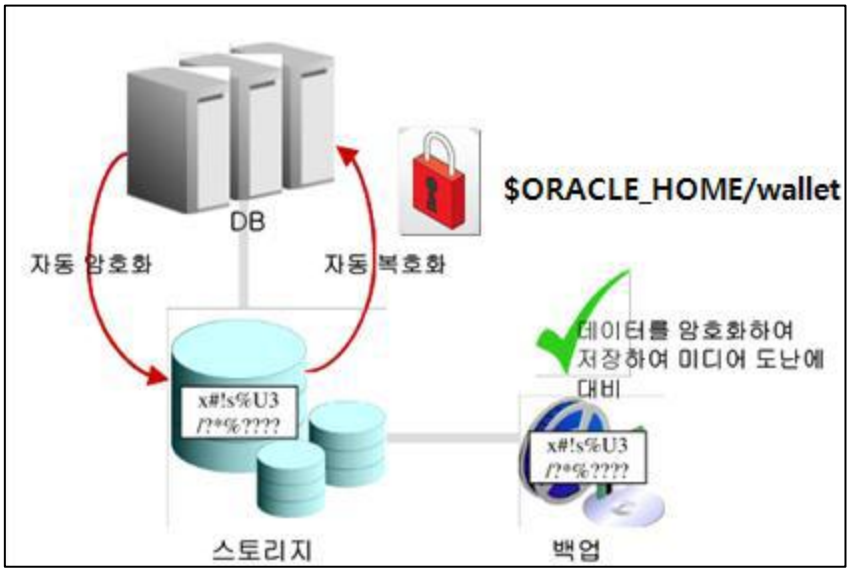
Version	변경일자	변경자(작성자)	주요내용
1.0	2011.12.05	박제헌 대리	문서 최초 작성
1.1	2012-02-02	박제헌 대리	Chapter OWM 사용법 추가
1.4	2012-02-03	박제헌 대리	메뉴편집
2.0	2013-04-12	고요한 대리 정관호 대리	각 환경에서의 Configuration 방법 추가. TDE 제약사항 추가. TDE 관련 detail 한 설명 추가.

Goodus
Breathing Life into Technology

Contents

1. 개요

TDE(Transparent Database Encryption)는 오라클 10gR2 의 New feature 이며 Enterprise Edition 에서 사용 할 수 있다. 테이블의 컬럼 데이터를 암호화 하여 디스크에 저장하는 기법 인데 디스크의 도난이나 백업 데이터 유출 시 데이터를 보호 할 수 있다. 디스크에 저장되어 있을 때에는 암호화 되어 있지만 메모리에 데이터가 올라올 때는 마스터 키를 이용하여 자동으로 복호화가 되어 사용자에게 보여지며 디스크에 저장 될 때에는 다시 암호화되어 저장되기 때문에 테이블에 적절한 권한이 있는 사용자는 TDE 를 적용 하더라도 암호화 되지 않은 데이터를 볼 수 있다. 10g 에서는 컬럼단위의 암호화를 지원하며 11g 에는 테이블 스페이스 단위의 암호화를 지원한다. 11g 에서는 테이블 스페이스 단위로 지원 되기 때문에 테이블, 클러스터, 인덱스, Lobs, 테이블, 인덱스 파티션도 암호화를 할 수 있다. TDE 는 타 솔루션과 달리 컬럼 단위와 테이블 스페이스 단위로 암호화 되기 때문에 application 의 변경이 필요하지 않는 것이 가장 큰 장점이라 할 수 있다.



1.1. Wallet File

Wallet File은 encryption과 decryption을 수행하는 마스터 키를 저장하고 있으며, wallet File은 데이터 베이스와 별도의 공간에 저장하여 허가된 사용자만 접근토록 한다.

$ORACLE_HOME/network/admin/sqlnet.ora 파일에 명시적으로 wallet File 의 위치를 지정 할 수 있다.

```
ENCRYPTION_WALLET_LOCATION=
        (SOURCE=(METHOD=FILE)(METHOD_DATA= (DIRECTORY= /oracle/product/admin/PROD/WALLET)))
```

위치를 지정하지 않았을 때는 기본적으로 $ORACLE_BASE/ADMIN/<SID>/WALLET 에 생성 되지만 경로가 존재하지 않거나 Platform 에 따라 파일 위치를 지정하지 않았을 때에는 ORA-28368 에러가 발생 하기도 한다.(Bug:4956266)

1.2. Wallet 위치 조회와 Wallet 상태를 조회

```
SQL> select wrl_parameter ,status from v$encryption_wallet;
WRL_PARAMETER                                    STATUS
------------------------------------------------ ------------------
/oracle11g/product/admin/ORCL/WALLET             OPEN
```

1.3. Encryption Key

① Master Encryption Key

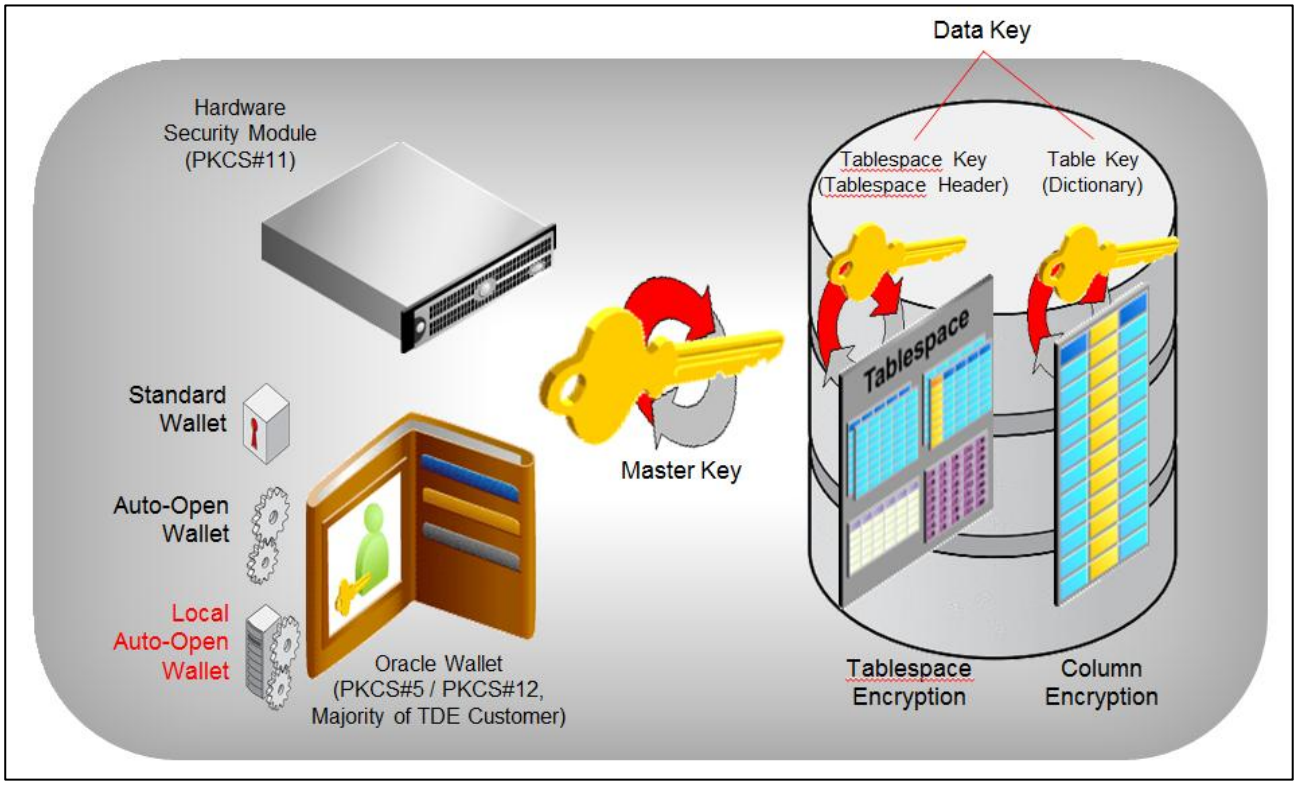
Master Key 는 column 또는 tablespace 에 사용되는 보조키를 암호화 하는데 사용되는 암호화 키이다.

② Table Key (column 단위-10g)

데이터 베이스 안의 Data Dicionary 에 저장되며 암호화된 테이블에 대하여 master key 와 함께 데이터를 암복호화 하는데 쓰인다.

③ Tablespace Key (tablespace 단위-11g)

11g 부터는 Encryption Key 가 각 tablespace header 안의 저장이 되어 데이터 암복호화에 사용이 된다.



1.4. TDE의 사용되는 암호화 Algorithms

암호화에 대한 국제 표준을 준수하는 Algotithms 을 사용하고 있다.

Algorithm	Key Size	Parameter Name
Triple DES (Data Encryption Standard)	168 bits	3DES168
AES (Advanced Encryption Standard)	128 bits (11gR2 default - tablespace)	AES128
AES	192 bits (10g default - Column)	AES192
AES	256 bits	AES256

2. TDE 적용 시 주의점

2.1. Password 관리

Wallet 이나 Wallet Password 를 분실하게 되면 암호화된 데이터는 Access 가 불가 하므로 물리적인 Wallet File 의 백업과 Wallet Password 관리가 반드시 필요하다.

2.2. Performance OverHead

2.2.1. CPU

① 11g : Tablespace 암호화는 Application 마다 다를 수 있지만, 평균적으로 5~8%사이에 Overhead 가 발생한다.(하지만 실무에서의 전체적인 Performance 를 본다면 11g 에서의 성능이 10g 보다 월등한 Performance 를 보여준다.)

② 10g : Column 암호화에서의 Table 은 Column 암호화가 삽입되어 있는 경우에만 해당이 되며, 암호화, 복호화 하는 과정에서 5%의 overhead 가 발생한다. 또한 암호화 된 column 수나 access 를 하는 횟수에 비례하기도 한다.

2.2.2. Large table Encryption

Size 가 큰 table 의 관하여 Column 암호화를 하는 경우 redo log size 를 증가시켜주어야 하고, 인덱스가 걸려 있는 상태에서의 Column 암호화는 상당한 시간이 소요된다.
그렇기 때문에 Index 를 제거후에 No Salt 옵션과 함께 암호화를 한 후, Index 를 새로 만들어 주어야 한다.(새로 만들어진 Index 는 암호화 된 값의 의해서 생성이 됨)

2.2.3. Storage

① 11g : Tablespace 암호화는 Storage OverHead 가 없다.

② 10g : Column 암호화는 일반 데이터보다 공간을 좀더 사용하게 되는데, AES 경우 최대 16byte,3DES 일 경우 8Bytes 가 추가적으로 사용되며 정합성 체크를 위한 20bytes 가 추가적으로 더 필요 하다.(nomac 파라미터 이용시 제외) salt 옵션으로 암호화 했을 경우 16bytes 가 더필요하여 결론적으로 최대 발생할 수 있는 스토리지 오버해드는 각 암호화 된 value 당 52bytes 가 된다.

2.3. 제약사항.

동일한 서버에 여러 데이터베이스가 설치되어 있는 경우, 각각의 Encryption Key 를 생성 및 사용해야 한다. 서로 공유하여 사용할 경우 암호화 된 데이터 손실을 유발할수 있기 때문에 Oracle 에서는 이 같은 방법은 지원하지 않는다.

2.3.1. Column Encryption 제약(10g ~ 11g) [ID 317311.1]

① B-tree 가 아닌 Index 유형(Function Index, Domain Index, Join Index 등등) 생성 불가.

```
SQL> create bitmap index scott.i_tde on scott.tde(c);
            create bitmap index scott.i_tde on scott.tde(c)
                                               *
            ERROR at line 1:
            ORA-28337: the specified index may not be defined on an
                       encrypted column
```

② Index 를 통한 Range Scan 불가능.(동등 비교 연산자를 통한 Scan 만 가능)

```
SQL> select * from  scott.emp2 where empno > 7900                    empno -> pk
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0          4          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0          4          0           2

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: SYS
Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         2          2          2  TABLE ACCESS FULL EMP2 (cr=4 pr=0 pw=0 time=2720 us cost=3 size=24 card=2)
```

★ Range Scan 은 불가..

```
SQL> select * from scott.emp2 where empno = 7900;

SQL ID: fkthsmpw58bb2 Plan Hash: 2129037341

select *
from
scott.emp2 where empno = 7900


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0          2          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: SYS
Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  TABLE ACCESS BY INDEX ROWID EMP2 (cr=2 pr=0 pw=0 time=5472 us cost=1 size=121 card=1)
         1          1          1   INDEX UNIQUE SCAN PK_EMP2 (cr=1 pr=0 pw=0 time=5453 us cost=0 size=0 card=1)(object
id 112810)★
```

③ External Table(BFILE) - oracle_datapump drver 로만 가능.

```
SQL> CREATE TABLE scott.tde_extract (c encrypt, c2)
        ORGANIZATION EXTERNAL
            (TYPE ORACLE_DATAPUMP
            DEFAULT DIRECTORY d_dir
            LOCATION ('tde.extract')
            )
        reject limit unlimited
        AS
        select * from scott.tde;

    Table created.

SQL> select * from scott.tde_extract;
          C         C2
  ---------- ----------
          1        100

SQL> select * from DBA_ENCRYPTED_COLUMNS;
 OWNER  TABLE_NAME  COLUMN_NAME ENCRYPTION_ALG    SAL
```

```
------ ----------- ----------- ----------------  ---
 SCOTT  TDE_EXTRACT C           AES 192 bits key  YES
```

④ TTS(Transportable Tablespace) 지원 불가.

```
$ expdp system/manager transport_tablespaces=users  directory=d_dir dumpfile=filefull.dmp
        encryption_password="x"

       Export: Release 10.2.0.0.0 - Beta on Tuesday, 14 June, 2005 16:02:08

       Copyright (c) 2003, Oracle.  All rights reserved.

       Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.0.0 - Beta
       With the Partitioning, Oracle Label Security, OLAP and Data Mining options
       ORA-39005: inconsistent arguments
       ORA-39032: function ENCRYPTION_PASSWORD is not supported in TRANSPORTABLE jobs
```

⑤ PK를 참조하는 FK키 Column 지원불가(ORA-28335)

```
SQL> create table scott.tde_fk (c number references scott.tde(c));
             create table scott.tde_fk (c number references scott.tde(c))
                                                            *
             ERROR at line 1:
             ORA-28335: referenced or referencing FK constraint column
                        cannot be encrypted
```

⑥ 암호화 Column을 Partition Key, Cluster Key로 사용할 수 없음.

```
SQL> create table SCOTT.TDE_PART
          (product_id  number(5) ENCRYPT,
           time_id     date)
           partition by range (product_id)
           (partition p1 values less than (500),
           partition p2 values less than (maxvalue))  ;

           partition by range (product_id)
                              *
    ERROR at line 4:
    ORA-28346: an encrypted column cannot serve as a partitioning column
```

⑦ LOB(BLOB/CLOB) 지원 불가

⑧ SYS 스키마 Object Column (ORA-28336) 지원불가

⑨ Export/Import Util (Expdp/Impdp 사용권장)STREAMS 사용불가.

⑩ Advanced Queuing 사용불가.

⑪ Advanced Replication 사용불가.

⑫ Logical Standby Database 사용불가.

⑬ Logminer에 의한 해석은 가능하지만, 암호화 Column은 “Unsupported Type”라고 표시됨.

⑭ 비 스칼라 Type (오브젝트 Type, 유저정의 Type, Varray, REF 등) 지원 불가

⑮ 암호화시 Salt 옵션을 부여할 경우 Index 생성 불가능

2.3.2. Tablespace Encryption 제약 및 변경사항. (11gR1~) [ID 432776.1]

① 모든 table, cluster, index, LOB, table 및 index partition 등을 지원가능.

② SYSTEM, SYSAUX, UNDO, TEMP tablespaces 는 암호화 불가능.

```
SQL> create undo tablespace TDE_UNDO ENCRYPTION USING '3DES168'
     default storage (ENCRYPT) datafile '/amer/rdbms/32bit/app/oracle/oradata/AMa111U4/tdeundo01.dbf' size 2M;

     ERROR at line 1:
     ORA-30024: Invalid specification for CREATE UNDO TABLESPACE

SQL> create temporary tablespace TDE_TEMP ENCRYPTION USING '3DES168'
     default storage (ENCRYPT) tempfile '/amer/rdbms/32bit/app/oracle/oradata/AMa111U4/tdetemp01.dbf' size 2M;

     create temporary tablespace TDE_TEMP ENCRYPTION USING '3DES168'
     *
     ERROR at line 1:
     ORA-25139: invalid option for CREATE TEMPORARY TABLESPACE
```

③ Exp 불가능, Imp 는 부분적으로 가능.(Expdp/Impdp 권장)

암호화가 설정이 안되어 있는 10gR2 또는 이전버전에서 Export 이후, 암호화 설정이 되어있는 11g 로 Import 는 가능하다.

```
$ exp system/manager tables='SCOTT.TDE1'

Export: Release 11.1.0.6.0 - Production on Fri Sep 21 17:10:16 2007

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security and Real Application Testing options
Export done in US7ASCII character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)

About to export specified tables via Conventional Path ...
Current user changed to SCOTT
EXP-00111: Table TDE1 resides in an Encrypted Tablespace TDE_TBS and will not be exported
Export terminated successfully with warnings.
```

```
$ expdp system/manager ENCRYPTION_MODE=password ENCRYPTION_PASSWORD=reencryptionpassword tablespaces=TDE_TBS

Export: Release 11.1.0.6.0 - Production on Friday, 21 September, 2007 17:11:39

Copyright (c) 2003, 2007, Oracle. All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security and Real Application Testing options
Starting "SYSTEM"."SYS_EXPORT_TABLESPACE_01": system/******** ENCRYPTION_MODE=password encryption_password=********
tablespaces=TDE_TBS
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
. . exported "SCOTT"."TDE1" 5.015 KB 1 rows
Master table "SYSTEM"."SYS_EXPORT_TABLESPACE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TABLESPACE_01 is:
/u01/app/oracle/admin/v1110/dpdump/expdat.dmp
Job "SYSTEM"."SYS_EXPORT_TABLESPACE_01" successfully completed at 17:12:25
```

④ Bitmap Index 사용 가능

```
SQL> create bitmap index scott.i_tde on scott.tde1(c);
Index created
```

⑤ Index 를 통한 Range Scan 가능

```
SQL> create index scott.i_tde2 on scott.tde1(c);
Index created.

SQL> select * from scott.tde1 where c between 1 and 3;
1  ...

3
768 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1398846556

-------------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       |   768 |  9984 |     3   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| I_TDE |   768 |  9984 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
    1 - access("C">=1 AND "C"<=3)
```

⑥ LOB datatype 가능

```
SQL> create table scott.tde_lob  (c clob) tablespace tde_tbs;
```

⑦ TTS 가능

-- Source 쪽에서 Wallet Key 를 Copy 해 가져가야하기 때문에 Target database 에서는 TDE 를 사용하지 않고 있어야만 가능하다. (Cross-endianism 은 불가능)

3. TDE Configuration

TDE 구성에 있어 크게 일반적인 Oracle 데이터 이관 기능(datapump, exp/imp, CTAS, alter table move, 기타..)을 이용하는 오프라인 방식과 Online Table Redefinition 기능을 이용한 온라인 방식으로 나눌수가 있다.

① Column 단위 : 암호화 대상 column 이 명확하고, 그 수가 많지 않을 때 사용을 한다.

② Tablespace 단위 : 암호화 대상 column 선정이 쉽지 않고, 그 수가 많을 때 사용한다.

3.1. Master Key Setting/resetting

처음 TDE 를 구성할 시 Master Key 를 생성해야하고, Database 를 마이그레이션 하거나, Clone DB, 업그레이드를 할 경우에는 Wallet 의 사본으로 하여금 새롭게 Master Key 를 resetting(권장)을 해야 한다.
또한 Setting/resetting 의 있어 wallet location 의 auto-login wallet 이 존재한다면 새로운 wallet 을 생성하지 않는다.

```
SQL> ALTER SYSTEM SET ENCRYPTION KEY ["certificate_ID"] IDENTIFIED BY "password"      → wallet setting/resetting
SQL> alter system set wallet open identified by  “password” ;          → wallet open
SQL> alter system set wallet close identified by  “password” ;         → wallet close
SQL> alter system set wallet close;                                    → auto-login wallet 사용시 wallet close
```

3.2. Silngle Instance Configuration

3.2.1. Wallet File 위치 지정

Wallet 파일이 지정될 위치를 sqlnet.ora 에 지정 한다.
Sqlnet.ora 파일에 wallet 의 경로를 지정 후에는 v$encryption_wallet 뷰를 통하여 sqlnet.ora 에 지정된 경로와 동일 하게 설정 되어 있는지 확인 한다.

```
$> vi $ORACLE_HOME/network/admin/sqlnet.ora
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=(METHOD=FILE)(METHOD_DATA=
    (DIRECTORY=/oracle/wallet1/)))
```

➔ Wallet File 의 default 위치는 $ORACLE_BASE/admin/<global_db_name>/wallet 으로 리눅스 환경에서는 명시적으로 위치를 지정하지 않을 경우 ORA-28368: cannot auto-create wallet 이 발생한다.

```
SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------- -------------------------------------- ------------------
file                  /oracle/wallet1/                       CLOSED
```

3.2.2. Master Key 생성

Master Key는 아래 명령어를 통하여 생성이 가능하며 최초 생성 시에만 사용해야 하며 아래의 문장으로 패스워드를 변경 할 경우에는 기존의 암호화 되었던 테이블을 사용할 수 없게 된다.
Wallet file 이름은 기본적으로 ewallet.p12로 생성이 된다.

```
SQL> alter system set encryption key identified by "oracle";
System altered.

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------- -------------------------------------- ------------------
file                  /oracle/wallet1/                       OPEN

$> ls -alrt /oracle/wallet1/
-rw------- 1 oracle dba 1309 Mar 7 05:14 ewallet.p12
```

3.2.3. Master Key OPEN 및 Close

데이터 베이스를 재 시작 할때마다 master key 를 아래의 명령어로 LOAD 해야 하며, 패스워드가 틀린 경우 ORA-28353: Failed to open wallet 에러가 발생한다.
또한 wallet 을 open 하지 않고 암호화된 Table 이나 Tablespace 를 조회 할때는 ORA-28365: wallet is not open 에러가 발생 한다.

혹 Master Key 를 Close 시에는 Auto-login 이 설정 되어있을 경우 패스워드 부분은 생략을 하여도 된다.

```
SQL> alter system set wallet open identified by "oracle";
System altered.

SQL> alter system set wallet close;      → wallet close
ORA-28390: auto login wallet not open but encryption wallet may be open  → auto-login 이 설정이 안되어서 발생.

SQL> alter system set wallet close identified by "oracle";
System altered.
```

3.3. 다중 Instance Configuration

싱글 Instance 와는 달리 서버 하나의 다중 Instance 구성이라면 TDE 구성의 있어 경로 설정방법은 다르다.
SQLNET.ORA 파일에 경로 설정에 있어 각 인스턴스는 SID 로 구분이 되고, 환경변수를 통하여 경로를 설정 하여야만 한다. [ID 1240824.1]

3.3.1. Wallet File 위치 지정

```
$> echo $ORACLE_SID
orcl1

$> vi $ORACLE_HOME/network/admin/sqlnet.ora
ENCRYPTION_WALLET_LOCATION =
(SOURCE =
(METHOD = FILE)
(METHOD_DATA =
(DIRECTORY = /oracle/wallet1/$ORACLE_SID)))

$> mkdir -p /oracle/wallet1/orcl1
```

3.3.2. Master Key 생성 및 OPEN

```
SQL> alter system set encryption key identified by "oracle";
System altered.

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------- -------------------------------------- ------------------
file                  /oracle/wallet1/$ORACLE_SID           OPEN

$> ls -alrt /oracle/wallet1/orcl1
-rw------- 1 oracle dba 1309 Mar 7 05:14 ewallet.p12

SQL> alter system set wallet open identified by "oracle";
System altered.
```

3.4. RAC 환경에서의 Configuration

RAC 환경에서의 wallet 관리는 자동화 되지 않고, Open, Close, modify 같은 작업은 각 Local 별로 수행을 해주어야 한다. 그렇기에 oracle 에서는 11g 부터 wallet file 을 shared storage 나 ACFS(ASM)의 저장하는 것을 권장을 하고 있다.

3.4.1. 일반적인 RAC 환경

RAC 에서는 먼저 1 번노드에서 Wallet 과 Master Key 를 생성 후 2 번노드로 Copy 를 하여 reopen 을 한 후 사용을 해야하고, 2 번 노드로 copy 를 한 후에는 wallet 을 재오픈을 해주어야 한다. [ID 567287.1]
(11g 에서도 마찬가지로 oracle 유저의 ORACLE_HOME/network/admin/ 경로에 sqlnet.ora 파일을 생성하면 된다.)

1 번노드

```
$> vi $ORACLE_HOME/network/admin/sqlnet.ora
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=(METHOD=FILE)(METHOD_DATA=
    (DIRECTORY=/oracle/wallet1/)))


SQL> alter system set encryption key identified by "oracle";

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------- -------------------------------------- ------------------
file                  /oracle/wallet1/                      OPEN
```

➔ 1 번노드에서 생긴 wallet file 을 2 번 노드로 SQLNET.ORA 파일에 기입할 경로로 복사한다.

2번 노드

```
$> vi $ORACLE_HOME/network/admin/sqlnet.ora
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=(METHOD=FILE)(METHOD_DATA=
    (DIRECTORY=/oracle/wallet1/)))

SQL> alter system set wallet close identified by “oracle” ;
System altered.

SQL> alter system set wallet open identified by "oracle";
System altered.


SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------  -------------------------------------  ------------------
file                  /oracle/wallet1/                       OPEN
```

3.4.2. RAC 환경에서의 다중 Instance Configuration

RAC 환경에서의 동일한 ORACLE_HOME (Engine)을 사용하고 하나이상의 database를 사용하는 경우에는 Unqname을 지정하여 Dynamic하게 구성하는 것을 권장하고 있다.

```
$> mkdir -p /oracle/wallets/orcl
$> vi sqlnet.ora
ENCRYPTION_WALLET_LOCATION =
 (SOURCE=(METHOD=FILE)
   (METHOD_DATA =
      (DIRECTORY=/oracle/wallets/$ORACLE_UNQNAME/)))

$> srvctl setenv database -d orcl -T  “ORACLE_UNQNAME=orcl”
```

3.4.3. RAC TDE 구성 시 고려사항.

① 각 인스턴스의 SQLNET.ORA 파일은 wallet 경로가 기입되어 있어야 한다.
② 하나의 인스턴스에서 Master Key를 생성 및 설정을 해주어야 하고, 그 이후에 다른 노드로 copy를 해야 한다.
③ 다른 노드로 복사가 완료 되었다면 모든 인스턴스에서 wallet을 재오픈을 해주어야 한다.
④ Master Key의 대한 변경 및 설정시에는 Wallet을 Open 하거나 Close를 하면 안된다.
⑤ Master Key의 대한 변경 및 설정시에는 모든 TDE 작업을 수행되어선 안된다.
⑥ Wallet Open 또는 Close시에는 RAC 모든 인스턴스에서 Open, Close를 해야한다.

4. Tool을 이용한 Wallet 설정

4.1. Orapki Tool

4.1.1. Wallet 생성

Orapki를 생성시 Password를 미리 기입할 수 있지만, 패스워드는 미리 기입안하는 것을 권장을 하고 있다.

➔ orapki wallet create -wallet wallet경로 [-pwd “wallet password”]

```
$> orapki wallet create -wallet /oracle/wallet/
Oracle PKI Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.
```

```
Enter password: oracle123
Enter password again: oracle123

$> ls /oracle/wallet/
ewallet.p12
```

4.1.2. Orapki를 이용한 Auto-login 설정.

데이터베이스를 재시작할 때마다 Master Key 를 수동으로 open 을 하는 번거러움을 해소하기 위하여 자동으로 open 을 해주는 설정을 할 수가 있다. [ID 445147.1]
하지만, 10g 의 경우 auto-login 을 한 후 Instance restart 를 하여도 상태는 CLOSE 가 나온다. [ID 1295713.1]
이는 해당 암호화 Table 에 대해 DDL 이나 DML 을 한 후에 상태를 체크해보면 wallet 상태는 다시 OPEN 으로 변경이 된다. 또한 wallet 상태를 Close 로 변경이 되더라도 암호화 Table 을 조회시 자동 Open 이 되어 조회가 가능해진다. 11g 에서는 DB restart 를 하여도 OPEN 상태가 유지가 되고 wallet 을 Close 를 하여도 VIEW 에서는 OPEN 상태로 표시가 된다.
개인적으로는 Auto-login 기능이 편할수도 있지만, 후에 이러한 기능으로 wallet 이 open, close 가 잘 안되는 문제가 발생 할 수도 있으며 오라클에서도 이 기능은 권장하지 않는다.

➔ orapki wallet create -wallet <wallet location> -auto_login -pwd "wallet password"

```
$> orapki wallet create -wallet /oracle/wallet1/ -auto_login -pwd "oracle"
SQL> create table test_col.test(col1 number, col2 varchar2(100) encrypt using 'AES256')
SQL> insert into test_col.test Values (1, '22' );
SQL> shutdown immediate
SQL> startup
SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                                STATUS
-------------------- -------------------------------------------- ------------------
file                  /oracle/wallet1/                             CLOSE

SQL> select * from test_col.test;
COL1           COL2
----------     ----------------
 1              22

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                                STATUS
-------------------- -------------------------------------------- ------------------
file                  /oracle/wallet1/                             OPEN


SQL> alter system set wallet close identified by "oracle";
ORA-28365: wallet is not open

SQL> alter system set wallet close;
System altered.

SQL> select * from test_col.test;
COL1           COL2
----------     ----------------
 1              22

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                                STATUS
-------------------- -------------------------------------------- ------------------
file                  /oracle/wallet1/                             OPEN
```

➔ wallet 을 Close 를 하여도 해당 암호화 Table 을 조회하면 wallet 상태는 자동 OPEN 이 된다.

➔ Wallet_location 의 PKCS#12(ewallet.p12)가 존재 한다면 기존 PKCS#12 wallet 에 대한 Password 를 제공해야한

다.

➔ Wallet_location 의 PKCS#12(ewallet.p12)이 존재하지 않다면 새로 wallet 이 생기면서 새로운 Password 를 지정해야 한다.

➔ Auto-login 기능을 제거시에는 OWM 을 이용하여 auto-login 기능을 해제해야 한다.

4.1.3. Orapki를 이용한 Password 변경(11g만 가능)

[ID 1193799.1]

orapki wallet change_pwd -wallet <path to the wallet> -oldpwd <oldpwd> -newpwd <newpwd>

패스워드 변경후에는 기존 Password 로 wallet 을 Close 한 후 새로운 Password 로 wallet 을 Open 을 해야 한다.

```
$> orapki wallet change_pwd -wallet /oracle/wallet/ -oldpwd "oracle" -newpwd "oracle123"
Oracle PKI Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.


SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------  -------------------------------------  ------------------
file                  /oracle/wallet1/                       OPEN

SQL> alter system set wallet close identified by "oracle";
System altered.

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------  -------------------------------------  ------------------
file                  /oracle/wallet1/                       CLOSE

SQL> alter system set encryption wallet open identified by "oracle";
alter system set encryption wallet open identified by "oracle"
*
ERROR at line 1:
ORA-28353: failed to open wallet


SQL> alter system set encryption wallet open identified by "oracle123";
System altered.

SQL> select * from v$encryption_wallet;
WRL_TYPE             WRL_PARAMETER                          STATUS
-------------------  -------------------------------------  ------------------
file                  /oracle/wallet1/                       OPEN
```

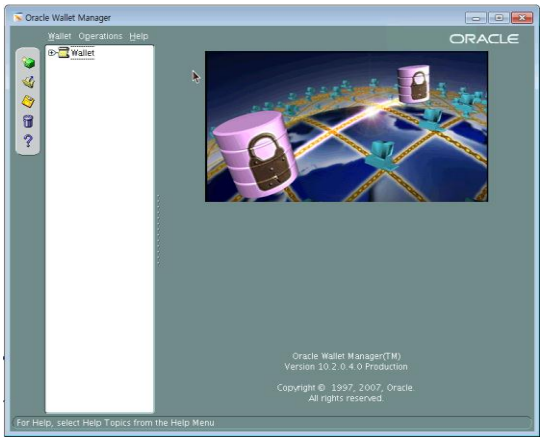
4.2. OWM

OWM 은 Oracle Wallet Management 로 wallet 파일을 보다 쉽게 관리하기 위한 GUI Tool 이다.
Master Key 생성, auto-login, 패스워드 변경과 같은 TDE 와 관련된 모든 것들을 이 Tool 하나로 관리를 할 수가 있다.

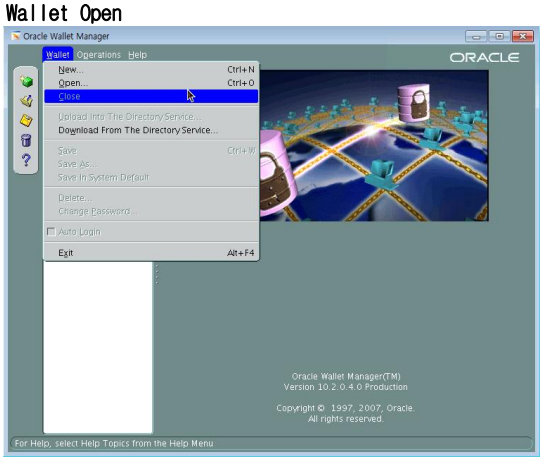
4.2.1. OWM을 이용한 Auto-login 설정

1. OWM 실행.

$> owm

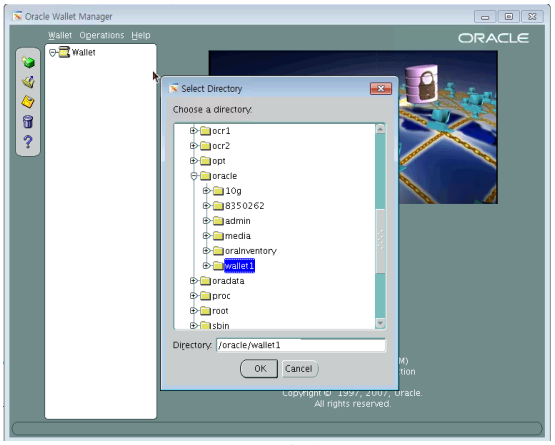


2. Wallet Open



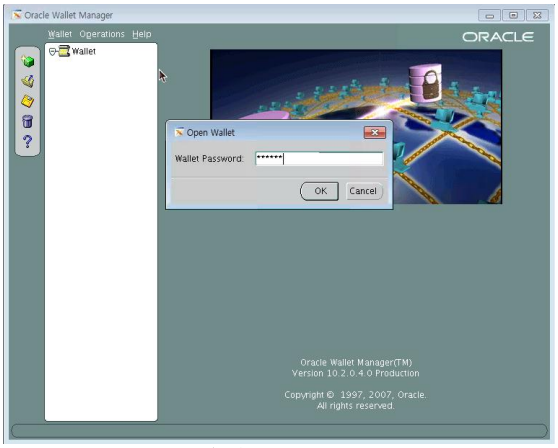
➔ 상단 wallet 메뉴 클릭
➔ Open 선택

3. Wallet 경로 설정



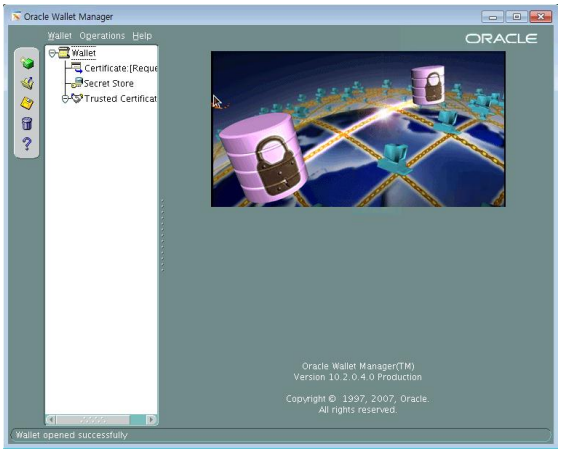
➔ Wallet 의 경로를 선택 후 'OK' 클릭.

4. Wallet 패스워드 입력.



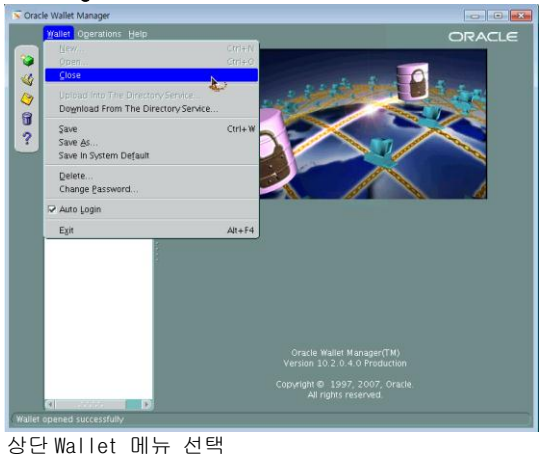
➔ 패스워드 입력 후 'OK' 클릭

5. Wallet File



➔ Wallet File 의 경로를 입력후에는 좌측 상단에 메뉴가 생김

6. Auto-login 설정



➔ 상단 Wallet 메뉴 선택
➔ Auto-login 체크
➔ Save 클릭

5. Table 암호화

5.1. Table 생성 시 Column을 암호화

Column 암호화시 'AES192' 는 default 값으로 using '3DES168' 를 작성을 안할시 기본적으로 AES192 가 적용이 된다.

```
SQL> create table test
    (first_name varchar2(11),
    last_name varchar2(10),
    order_number number(13),
    credit_card_number varchar2(20) encrypt using '3DES168' no salt 'NOMAC') ;

SQL> insert into test values ('Jon', 'Oldfield', 10001, '5446-9597-0881-2985');
SQL> insert into test values ('Chris', 'White', 10002, '5122-3580-4608-2560');
SQL> insert into test values ('Alan', 'Squire', 10003, '5595-9689-4375-7920');
```

5.1.1. Salt Option

암호화 시의 사용하는 Salt Option 은 데이터의 보안을 강화하는 방법으로 임의의 String 으로 암호화 하는 것이며 암호화 할 때 마다 다른 String 패턴으로 암호화가 진행이 된다.
Salt Option 를 사용하게 되면 추가적으로 각 데이터 값마다 16byte 가 필요하며, 인덱스를 사용할 예정이라면 Salt Option 을 사용해서는 안된다.

5.1.2. NOMAC Parameter

TDE 는 기본적으로 SHA-1 알고리즘을 사용하여 무결성을 체크 하지만, NOMAC 파라미터를 사용하게 되면 암호화 및 복호화시에 하는 무결성 체크부분이 빠지면서 TDE 의 관한 성능 오버헤드를 줄일 수 있다.
이 파라미터를 사용하게 되면 각 암호화 된 값보다 디스크 공간이 20Byte 가 더 필요하게 되고, 이미 SHA-1 알고리즘을 사용하여 Table Column 암호화가 있다면, 동일한 Table 의 다른 Column 을 NOMAC Parameter 로 암호화를 할 수 없다.

```
➔ 무결성 알고리즘 변경.
SQL> ALTER TABLE test REKEY USING '3DES168' 'SHA-1';
Table altered.

SQL> ALTER TABLE test REKEY USING '3DES168' 'NOMAC';
Table altered.
```

5.2. External Table 생성시 암호화

External Table 은 오직 DataPump 로 사용할 때 가능하고, External Table 을 새로운 위치로 Move 를 하게 되면 임의로 생성 된 Encryption Key 를 사용 못하게 된다. 그렇기에 컬럼 암호화시 따로 암호를 지정하여 External Table 이 Move 를 하더라도 엑세스를 가능하게 해주도록 해야한다.

```
SQL> create directory d_dir as '/oracle' ;
SQL> grant read, write on directory d_dir to hr;
SQL> conn hr/hr
SQL> create table test_ext
      (empno number(10),
      Empname varchar2(30),
      Salary number(10) ENCRYPT USING '3DES168' IDENTIFIED BY "oracle" )
      ORGANIZATION EXTERNAL
      (
      TYPE ORACLE_DATAPUMP
      DEFAULT DIRECTORY "D_DIR"
```

```
    LOCATION('test_ext.dat')
    )
```

5.3. 기존 TABLE의 암호화 Column을 추가 또는 변경시.

AES192 알고리즘과 Salt, MAC Option 은 default 값으로 Index 를 사용할 예정이라면 No salt Option 을 부여해야 한다.

5.3.1. Column 추가시

```
SQL> ALTER TABLE test ADD (sales VARCHAR2(11) ENCRYPT no salt 'NOMAC' );
```

5.3.2. Column 변경시

```
SQL> ALTER TABLE test MODIFY (sales ENCRYPT  no salt 'NOMAC' );
```

5.4. 암호화 Column을 제거시

```
SQL> ALTER TABLE test MODIFY (sales DECRYPT);
```

5.5. 암호화 Column의 대한 Encryption Key와 알고리즘 변경시.

5.5.1. Encryption Key 변경시

```
SQL> ALTER TABLE test REKEY;
```

5.5.2. 암호화 알고리즘 변경시

```
SQL> ALTER TABLE employee REKEY USING '3DES168';
```

6. Tablespace 암호화

11gR2 에서는 Column 암호화와 Tablespace 암호화 모두 동일한 Master Key 를 사용하고, 만약 10g 에서 11g 로 업그레이드 한것이라면 Master Key 는 재생성을 해야만 한다.(11gR1 에서 11gR2 로 업그레이드도 재생성이 필요하고 **ewallet.p12 파일도 그대로 가져가야 한다.)**
또한 Oracle 에서 Auto-login 기능은 권장하지 않으며, 기존 Tablespace 는 암호화 할 수 없기 때문에 새롭게 암호화 된 Tablespace 를 생성하여 CTAS, Table Move, DataPump 등의 방법으로 이동시켜야 한다.

6.1. Wallet Open

암호화 된 Tablespace 를 만들기 전에 Wallet 은 Open 이 되어있어야만 하고, Instance Crash 로 인한 recover 가 필요할 시 redo 와 Undo 의 암호화 된 Data 의 대한 접근이 필요하기 때문에 Database 는 Open 이 되기전의 Wallet 이 Open 되어 있어야만 한다.

```
SQL> startup mount;
SQL> alter system set encryption wallet open identified by “oracle” ;
SQL> alter database open;
```

6.2. 암호화 된 Tablespace 생성

Tablespace 암호화는 기본 알고리즘이 AES128 이다.

```
SQL> CREATE TABLESPACE test DATAFILE '/oradata/test01.dbf' SIZE 100M
     ENCRYPTION USING '3DES168'
     DEFAULT STORAGE(ENCRYPT);

SQL> Select tablespace_name,encrypted from dba_tablespaces where tablespace_name='TEST';
TABLESPACE_NAME                ENC
------------------------------ ---
TEST                           YES

SQL> Select * from v$encrypted_tablespaces;
TS#         ENCRYPT  ENC
----------- -------  ---
5           3DES168  YES
```

6.3. 암호화 Tablespace Decryption

Tablespace 암호화를 해제하기 위해서는 해당 Table 들을 alter table move, DataPump, dbms_redefinition 을 이용하여 암호화 되지 않은 Tablespace 로 옮기면 된다.

```
SQL> alter table cust_payment_info_2 move tablespace users;
Table altered.

SQL> select * from cust_payment_info_2;
FIRST_NAME  LAST_NAME  ORDER_NUMBER CREDIT_CARD_NUMBER
----------- ---------- ------------ --------------------
Jon         Oldfield   10001        5446-9597-0881-2985
Chris       White      10002        5122-3580-4608-2560
Alan        Squire     10003        5595-9689-4375-7920

SQL> alter system set wallet close identified by "oracle";
System altered.

SQL> select * from cust_payment_info_2;
FIRST_NAME  LAST_NAME  ORDER_NUMBER CREDIT_CARD_NUMBER
----------- ---------- ------------ --------------------
Jon         Oldfield   10001        5446-9597-0881-2985
Chris       White      10002        5122-3580-4608-2560
Alan        Squire     10003        5595-9689-4375-7920
```

7. Expdp/Impdp 사용법

TDE 로 컬럼 암호화를 하였을 경우 exp/imp 는 사용할 수 없다. 단 11g 에서는 export 덤프 파일을 암호화 된 Tablespace 로 Import 는 사용 할 수 있다.

7.1. Expdp/impdp

➔ test 라는 유저의 default tablespace 는 test 로 지정이 되어있고, test tablespace 는 암호화 된 tablespace 인 환경이다.

```
SQL> select ts#,  ENCRYPTIONALG  , ENCRYPTEDTS from  v$encrypted_tablespaces;
TS#                 ENCRYPT          ENC
----------          --------          ----
8                   3DES168          YES

SQL> conn test/oracle
SQL> select count(*) from test;
COUNT(*)
----------
10000
```

➔ 기존 exp/Imp 는 metadata 들은 정상적으로 수행이 되지만, 실제 데이터가 있는 Table 같은 경우는 Error 가 발생하면서 export 가 수행이 되지 않는다.

```
$> exp test/oracle file=/oracle/exp.dmp log=exp.log owner=test

Export: Release 11.2.0.3.0 - Production on Wed Apr 10 14:29:43 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Tes
Export done in KO16MSWIN949 character set and AL16UTF16 NCHAR character set
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user TEST
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user TEST
About to export TEST's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export TEST's tables via Conventional Path ...
EXP-00111: Table TEST resides in an Encrypted Tablespace TEST and will not be exported
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
```

```
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.
```

➔ Expdp/Impdp 를 사용하는 경우.
동일한 wallet File(Master Key)를 사용하고 있다면 Expdp/impdp 를 수행함에 있어 아무러 Error 는 발생하지 않는다. 하지만 Master Key 가 없거나 잘못 되어있을 경우에는 ORA-28365: Wallet is not open 이 발생한다.

```
$> impdp system/oracle directory=en dumpfile=expdp.dmp logfile=impdp.log schemas=test remap_schema=test:hr
Import: Release 11.2.0.3.0 - Production on Wed Apr 10 14:50:54 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
Master table "SYSTEM"."SYS_IMPORT_SCHEMA_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_SCHEMA_01":  system/******** directory=en dumpfile=expdp.dmp
logfile=impdp.logschemas=test remap_schema=test:hr
Procssing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "HR"."TEST"                                 151.6 KB   10000 rows
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Job "SYSTEM"."SYS_IMPORT_SCHEMA_01" completed with 0 error(s) at 14:50:57
```

7.2. Dump File 암호화

encryption_password="Password" 옵션을 이용하면 암호화 된 table 을 암호화 형식의 dump 파일로 받을 수 있다.

```
$> expdp test/oracle schemas=test directory=en dumpfile=expdp.dmp logfile=expdp.log encryption_password="oracle"

Export: Release 11.2.0.3.0 - Production on Wed Apr 10 15:48:43 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
Starting "TEST"."SYS_EXPORT_SCHEMA_01":  test/******** schemas=test directory=en dumpfile=expdp.dmp logfile=expdp.log
encryption_password=********
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 256 KB
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
. . exported "TEST"."TEST"                               151.7 KB   10000 rows
```

```
Master table "TEST"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TEST.SYS_EXPORT_SCHEMA_01 is:
  /oracle/expdp.dmp
  Job "TEST"."SYS_EXPORT_SCHEMA_01" successfully completed at 15:49:22



$> strings expdp.dmp |more
11.02.00.00.00
001:001:000001:000001
3xPf
fDFpQ
KAZMR
BD)C
>`z!
!>u1o
iWse*
GNIH
U=:+I
qvl&
XOOhF>"
4{1-
{d{W
'&<PW
.?0[
h0,p7on
fN*oA
GK.bn
A%3C
+nH6F
```

➔ 덤프파일에 strings으로 확인을 해보면 실제 값들은 임의의 값들로 암호화 되어 들어간 것을 알 수가 있다.

그 외에도 Encryption, Encryption_Algorithm, Encryption_Mode 옵션을 더 추가하여 좀 더 디테일하게 암호화를 할 수도 있다.