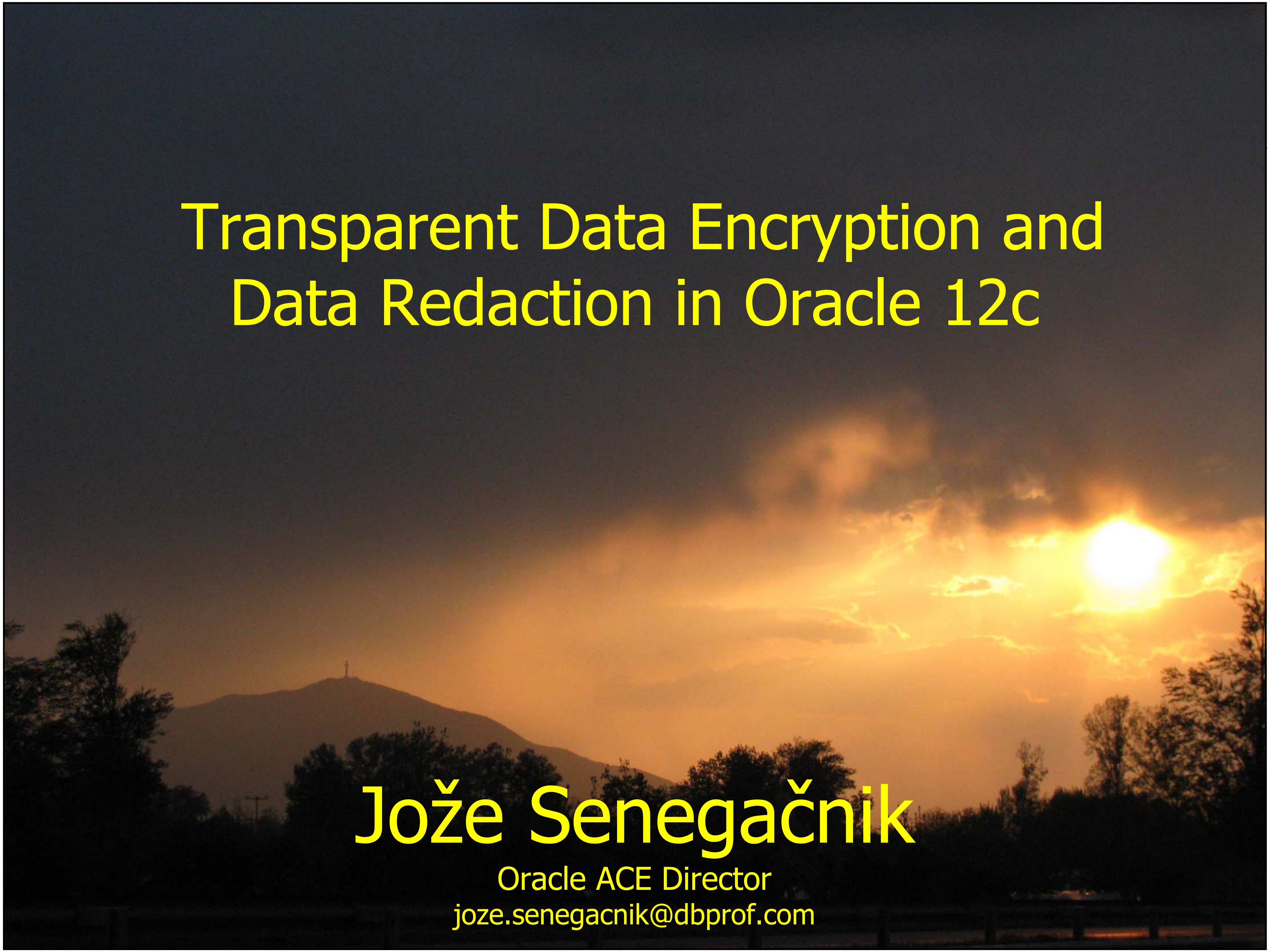


Transparent Data Encryption and Data Redaction in Oracle 12c

Jože Senegačnik

Oracle ACE Director

joze.senegacnik@dbprof.com

About the Speaker

Jože Senegačnik

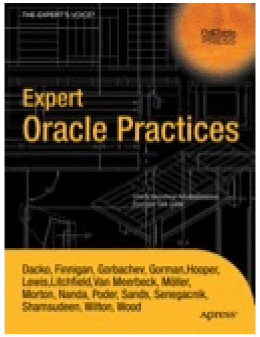
- First experience with Oracle Version 4 in 1988
- 27 years of experience with Oracle RDBMS.
- Proud member of the OakTable Network www.oaktable.net
- Oracle ACE Director
- Co-author of the OakTable book “Expert Oracle Practices” by Apress (Jan 2010)
- VP of Slovenian OUG (SIOUG) board
- CISA – Certified IS auditor
- Blog about Oracle: http://joze-senegacnik.blogspot.com

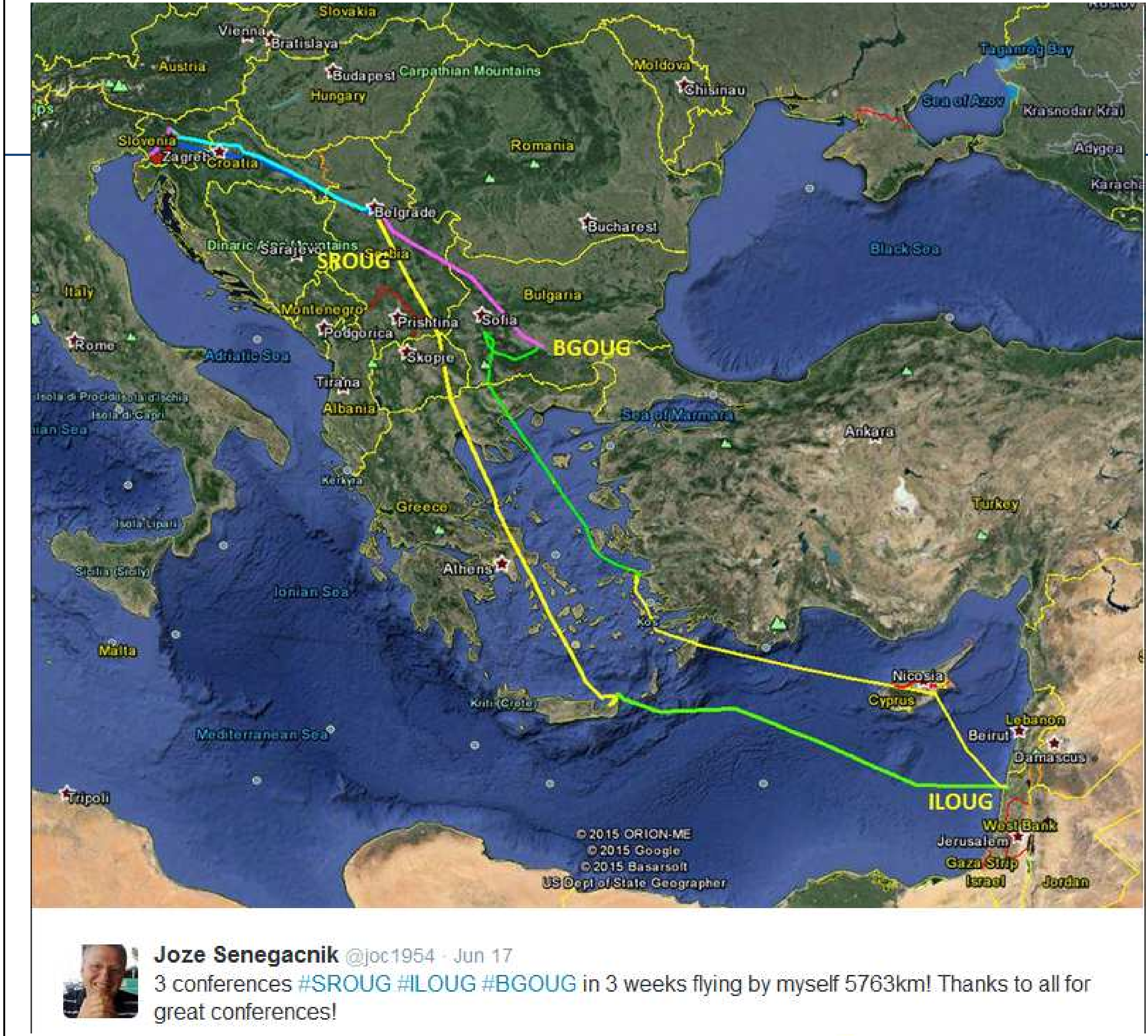
- PPL(A) – private pilot license PPL(A) / instrument rated IR/SE
- Blog about flying: http://jsenegacnik.blogspot.com
- Blog about Building Ovens, Baking and Cooking: http://senegacnik.blogspot.com

OakTable.net

ORACLE ACE Director

Oracle ACE


Expert
Oracle Practices
Dacko, Finnigan, Gorbachev, Gorman, Hooper, Lewis, Litchfield, Van Meerbeck, Möller, Morton, Nanda, Poder, Sands, Senegacnik, Shamsudeen, Wilton, Wood
Apress



Joze Senegacnik @joc1954 · Jun 17

3 conferences #SROUG #ILOUG #BGOUG in 3 weeks flying by myself 5763km! Thanks to all for great conferences!

SROUG - Serbia

ILOUG - Israel

BGOUG - Bulgaria

Introduction

- Transparent Data Encryption (TDE) resides within the database to prevent database bypass while still being transparent to applications and easy to deploy. (since Oracle10g)
- TDE encryipts data on data file level.
- TDE can encrypt:
  - entire application tablespaces
  - or just columns.
- TDE is transparent to applications – no need to any application changes

Introduction

- Application users do not deal directly with encrypted data.
- TDE has built-in two-tier encryption key management providing:
  - provides full key lifecycle management
  - tracking the keys across their lifetime
  - assisted encryption key rotation,
  - switch to a new master key with no downtime.
- Data encrypted with TDE tablespace encryption remains protected on backup media that could pose opportunities for bypass attacks.

Reduce Encryption impact

- Encryption facts:
  - Data is encrypted when the database block is written to the data file.
  - Data is decrypted when the database block is read from the data file into SGA/PGA
- Every physical read/write requires database block decryption/encryption.
- Database blocks in buffer cache are in decrypted form. They are accessed as any other database block.
- When block is changed (updated) the database writer process writes it to the datafile in encrypted form.

Two-tier Key Management

- Two-tier key management architecture consisting of:
  - data encryption keys and
  - master encryption key.
- The data encryption keys are:
  - Managed automatically
  - Are encrypted by the master encryption key.
- The master encryption key is stored and managed outside of the database within an Oracle Wallet (a standards-based PKCS12 file that protects keys).
- Master key should be kept separately from the encrypted data (especially when performing backup)
- The two-tier key architecture also enables rotation of master keys without having to re-encrypt all of the sensitive data.
- Oracle Database 12c introduces a new dedicated SYSKM role for optional delegation of all key management functions to a privileged user account including:
  - rotating master keys
  - changing password of the keystore.

TDE Glossary

- **Master encryption key** – The encryption key used to encrypt secondary data encryption keys used for column encryption and tablespace encryption. Master encryption keys are part of the Oracle Advanced Security two-tier key architecture.
- **Unified master key** – The unified master encryption key is generated with the first re-key operation in an Oracle Database 11g Release 2. The unified master key can be easily re-keyed (rotated).
- **Tablespace key** – The key used to encrypt a tablespace. These keys are encrypted using the master key and are stored in the tablespace header of the encrypted tablespace, as well as in the header of each operating system file that belongs to the encrypted tablespace.
- **Wallet** – A PKCS#12 formatted file outside of the database, encrypted based on password-based encryption as defined in PKCS#5. Used to store the TDE master key.
- **Advanced Encryption Standard (AES)** – A symmetric cipher algorithm defined in the Federal Information Processing (FIPS) standard no. 197. AES provides 3 approved key lengths: 256, 192, and 128 bits.

Key Generation

- Prerequisite for any encryption process is to create wallet what is done with the following command:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
'/u01/app/oracle/admin/<db_unique_name>/wallet' IDENTIFIED BY <password>;
```

- The database user that can do this is either member of DBA OS group or SYSKM OS Group.

- When the keystore is generated it is opened

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <password>;
```

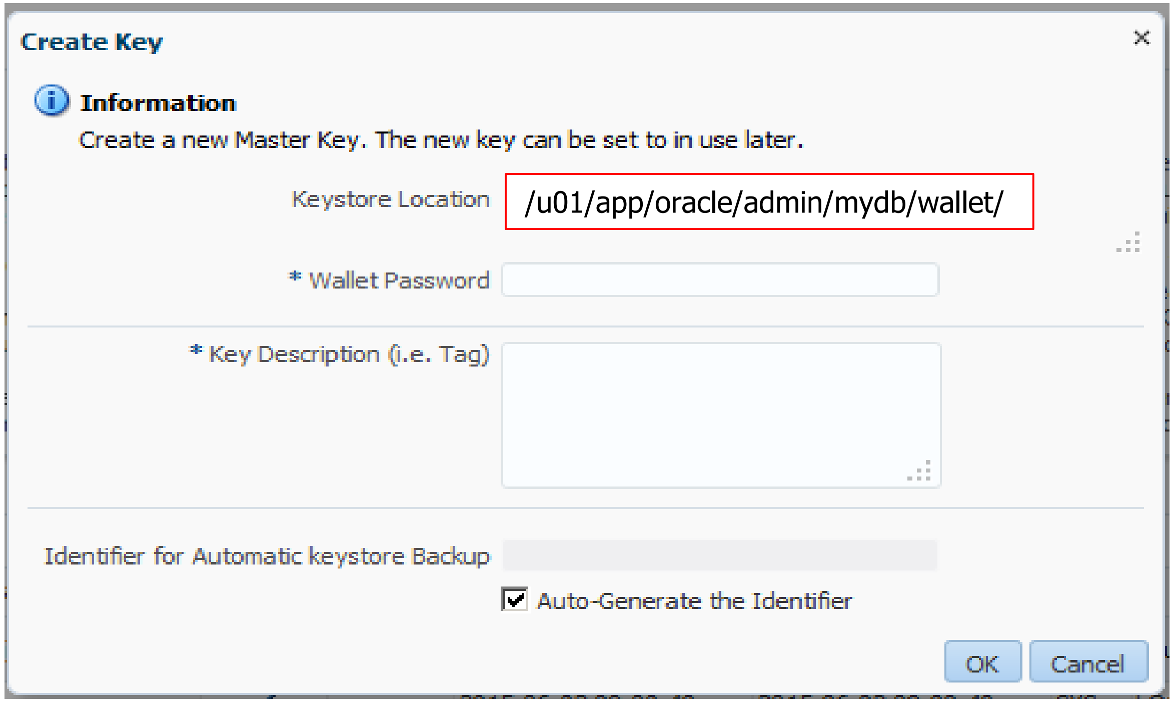
- The master key is generated for the first time using the following command:

```
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY USING TAG 'Y2015' IDENTIFIED
BY <password> WITH BACKUP;
```

- Actually the same command is used to rotate(rekey) the master key.

Generate Master Key in EM

- In Oracle Enterprise Manager the master key can be created via this entry form:



Create Key

Information

Create a new Master Key. The new key can be set to in use later.

Keystore Location /u01/app/oracle/admin/mydb/wallet/

* Wallet Password

* Key Description (i.e. Tag)

Identifier for Automatic keystore Backup

☑ Auto-Generate the Identifier

- The key custodians should enter wallet password and also define the key description tag which could be used to somehow describe the master key, for instance Y2015

Key Storage

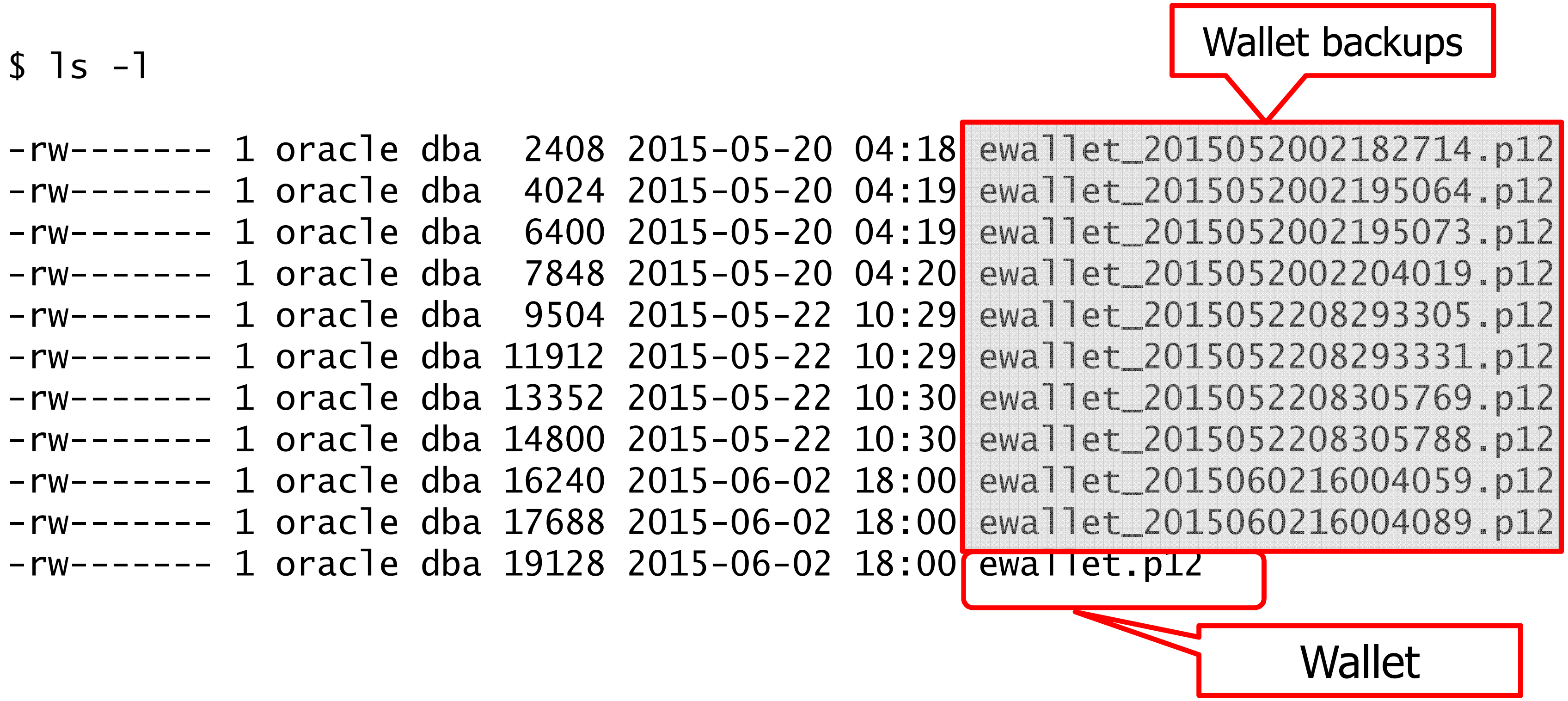
- The TDE master key, stored in the Oracle Wallet, is generated by Oracle during the initial configuration of TDE.
- The master key is generated using a pseudo-random number generator inside the Oracle database.
- The wallet is a critical component and should be backed up in a secure location, on-site and offsite.
- Backup the wallet associated with the master key:
  - immediately after it is initially created,
  - whenever the master key is changed,
  - before changing the wallet password.
- Location for storing wallets can be anywhere on the disk.
- The whole directory where wallet is stored should be part of regular system backup.
- Wallet should be always backed up separately from the database files / database backup.

Wallet Store Definition

- The wallet location is defined in sqlnet.ora file in $ORACLE_HOME/network/admin directory.
- The definition about the wallet in sqlnet.ora is like the following:

```
ENCRYPTION_WALLET_LOCATION=
 (SOURCE=
  (METHOD=FILE)
   (METHOD_DATA=
    (DIRECTORY=/u01/app/oracle/admin/$ORACLE_SID/wallet)))
```

Protection on Linux



```
$ ls -l

-rw------- 1 oracle dba  2408 2015-05-20 04:18 ewallet_2015052002182714.p12
-rw------- 1 oracle dba  4024 2015-05-20 04:19 ewallet_2015052002195064.p12
-rw------- 1 oracle dba  6400 2015-05-20 04:19 ewallet_2015052002195073.p12
-rw------- 1 oracle dba  7848 2015-05-20 04:20 ewallet_2015052002204019.p12
-rw------- 1 oracle dba  9504 2015-05-22 10:29 ewallet_2015052208293305.p12
-rw------- 1 oracle dba 11912 2015-05-22 10:29 ewallet_2015052208293331.p12
-rw------- 1 oracle dba 13352 2015-05-22 10:30 ewallet_2015052208305769.p12
-rw------- 1 oracle dba 14800 2015-05-22 10:30 ewallet_2015052208305788.p12
-rw------- 1 oracle dba 16240 2015-06-02 18:00 ewallet_2015060216004059.p12
-rw------- 1 oracle dba 17688 2015-06-02 18:00 ewallet_2015060216004089.p12
-rw------- 1 oracle dba 19128 2015-06-02 18:00 ewallet.p12
```

Wallet backups

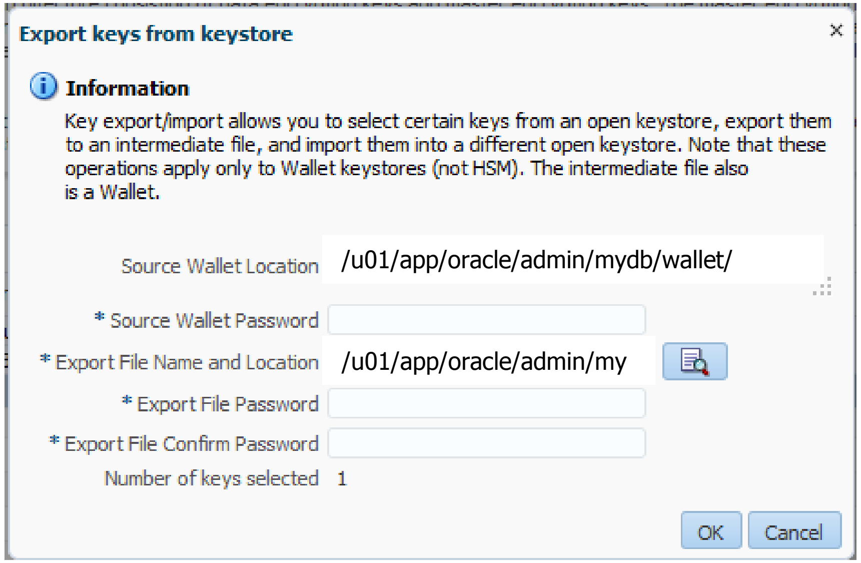
Wallet

Creating Encrypted Tablespace

- Different algorithms avaialable for encription
  - Some are more, some are less secure
  - Complex algorythms require more CPU power and have bigger performance penality
  - Keystire should be already opened before creating encrypting tablespaces
- CREATE TABLESPACE SENSITIVE_DATA DATAFILE '+DATA' SIZE 1G ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
- Tables can be moved inito encrypted tablespace with „ALTER TABLE MOVE“ command.

Exporting Encryption Keys

- Keys can be exported and imported from/to wallet.
- This can be done via Oracle Enterprise Manager



Export keys from keystore

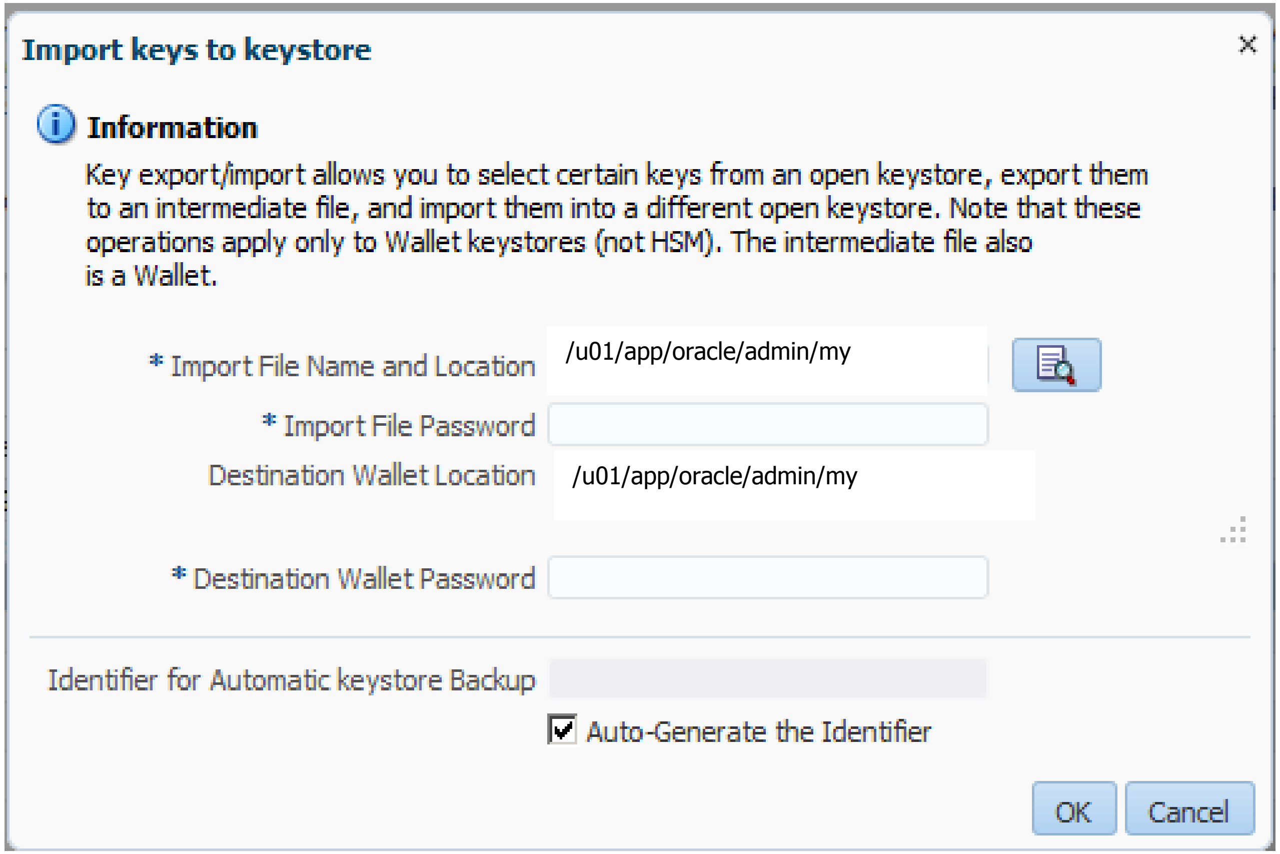
Information

Key export/import allows you to select certain keys from an open keystore, export them to an intermediate file, and import them into a different open keystore. Note that these operations apply only to Wallet keystores (not HSM). The intermediate file also is a Wallet.

Source Wallet Location	/u01/app/oracle/admin/mydb/wallet/
* Source Wallet Password	
* Export File Name and Location	/u01/app/oracle/admin/my
* Export File Password	
* Export File Confirm Password	
Number of keys selected	1

Importing Encryption Keys

- Key custodians need to enter import file password and destination wallet password.
- This action should be logged in key management log and signed off by key custodians



Import keys to keystore

Information

Key export/import allows you to select certain keys from an open keystore, export them to an intermediate file, and import them into a different open keystore. Note that these operations apply only to Wallet keystores (not HSM). The intermediate file also is a Wallet.

* Import File Name and Location: /u01/app/oracle/admin/my

* Import File Password:

Destination Wallet Location: /u01/app/oracle/admin/my

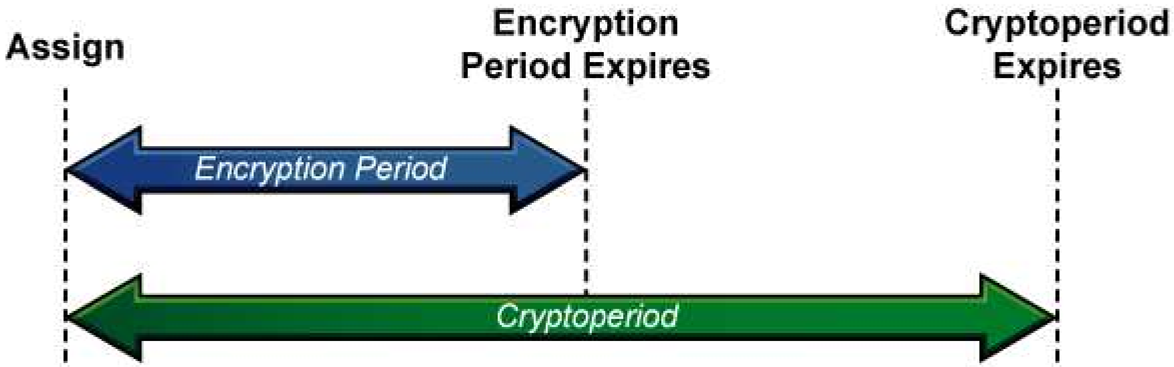
* Destination Wallet Password:

Identifier for Automatic keystore Backup:

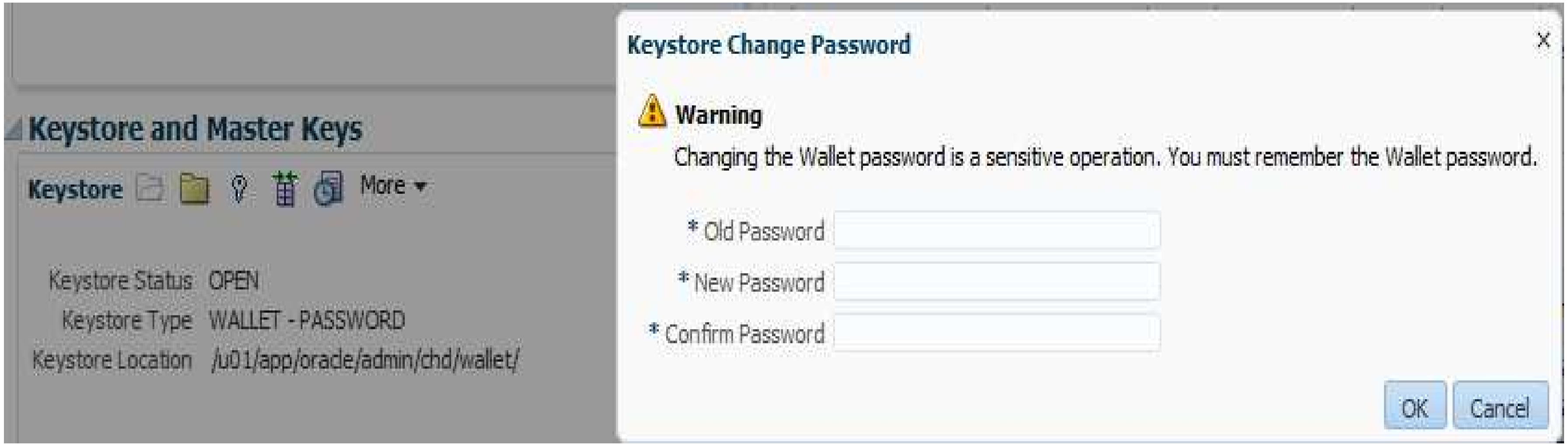
☑ Auto-Generate the Identifier

Key Lifecycle

- The key lifecycle is based on two time periods defined in the key policies:
  - Encryption period
  - Crypto period
- **Encryption** period is the period after a key is assigned that can be used to encrypt data.
- **Crypto period** is the period that can be used for decryption.
- The two periods start at the same time when the key is assigned.



Changing Wallet Password



Keystore and Master Keys

Keystore Status OPEN
Keystore Type WALLET - PASSWORD
Keystore Location /u01/app/oracle/admin/chd/wallet/

Keystore Change Password

Warning
Changing the Wallet password is a sensitive operation. You must remember the Wallet password.

* Old Password
* New Password
* Confirm Password

Applying Encryption in Oracle Multitenant Architecture

- Oracle Advanced Security fully supports Oracle Database 12c multitenant architecture.
- TDE automatically follow Pluggable Databases (PDB) as they move between multitenant container databases.
- When moving an encrypted PDB, the TDE master keys for that PDB are transferred separately from the encrypted data to maintain proper security separation during transit.
- Encryption immediately resume their normal operation after the PDB has been plugged in and configured.

Standby database/RAC

- Master key (wallet) is created only once (on one node or on primary database)
- It is copied to standby database / other RAC nodes (don‘t forget to enter definition of wallet location in sqlnet.ora)
- The password should be entered at database / instance startup
- Multitenant option requires additional steps to be performed during startup. Good news – they can be automated.

Oracle Restart/RAC/DataGuard

- If normal wallet is used after database restart the wallet password should be used.
- To enable automatic restart we need to create autologin wallet with *orapki* utility.
- This will enable automatic failover/switchover in Data Guard environment.
- We will be prompted to enter wallet password.

```
orapki wallet create -wallet /u01/app/oracle/admin/<db_unique_name>/wallet -
auto_login
```

Manual start of Multitentant Database

```
--STARTUP OF MULTITENANT DATABASE WITH TDE
startup mount

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
<password>;

alter database open;

alter session set container = mypdb;

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <password>
CONTAINER=CURRENT;
```

Data Redaction

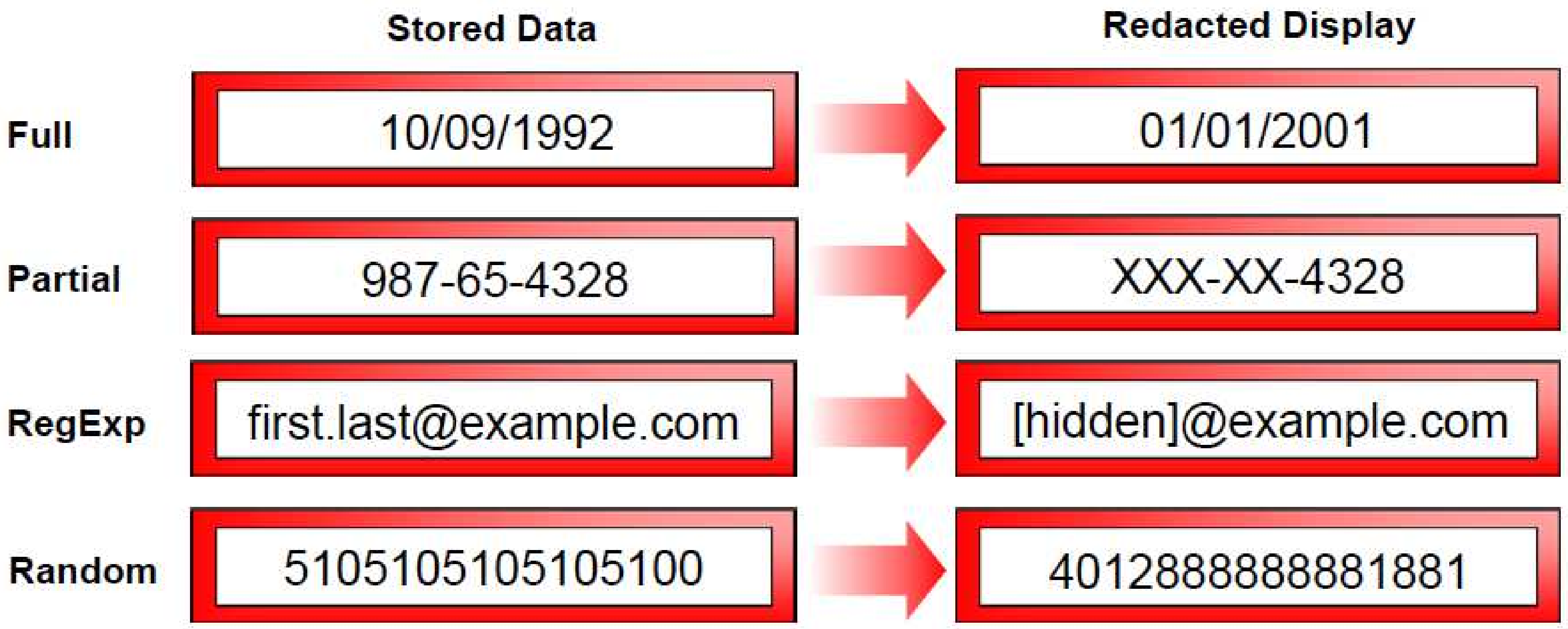
Data Redaction

- New feature in Oracle 12c licencesd under Advanced Security Option in EE.
- Esentially it is real-time data hiding / obscuration that is transparent to the applications.
- *Not to be confused with **Data Masking** (another option in EE) that changes the values of actual data.*
- No changes od data at phyiscal level, just obscuration at runtime.
- Data redaction is performed at query execution.
- It does not prevent priviliged users like DBAs to see data in original form.
- Governed by **Redaction Policies** that specify conditions that must be met that the data redation will be performed.
- Data redaction is performed only on the columns defined in the SELECT list.
  - No redation on columns used in WHERE clause
  - No redaction takes place in subqueries, only at the top level SELECT statement
  - Data redaction is not reported in the execution plan.

Types of Data Redaction

- **Full redaction** – Whole content of the column is protected.
  - Value returned depends on column data type:
    - numeric type columns will return 0 (zero)
    - character type columns will return space but this can be changed at database level.
- **Partial redaction** – Only part of the information is changed. Most common example would be hiding some of parts of credit card number (PAN) and replacing with "*"
- **Regular expressions** – replacement of certain patterns of data defined as regular expression.
- **Random redaction** – Actual data values are replaced with random values that change for each execution.
- **No redaction** – Used for testing purposes without affecting the results of queries.
  - Very usefull to test policies in advance and find potential problems that might appear later on in production.

Data Redaction Examples



	Stored Data	Redacted Display
Full	10/09/1992	01/01/2001
Partial	987-65-4328	XXX-XX-4328
RegExp	first.last@example.com	[hidden]@example.com
Random	5105105105105100	4012888888881881

New Package DBMS_REDACT

- Data redaction is implemented in database kernel and administered via DBMS_REDACT package.
- DBMS_REDACT.**ALTER_POLICY** - Allows changes to existing policies.
- DBMS_REDACT.**DISABLE_POLICY** - Disables an existing policy.
- DBMS_REDACT.**DROP_POLICY**- Drop an existing policy.
- DBMS_REDACT.**ENABLE_POLICY** - Enables an existing policy.
- DBMS_REDACT.**UPDATE_FULL_REDACTION_VALUE** - Change the default return value for full redaction.
  - The database **must be restarted** to take effect.

Full Data Redaction

- Character Data Types – Returned value is ' ' (single space)
- Number Data Types - Output text is a „0“ (zero)
- Date-Time Data Types - The output text is set to 1.1.2001

RANDOM Data Redaction

- CHAR Data Types - Redacted in same character set and byte length as the column definition

- Number Data Type - Redacted in same character set and the length is limited based on the length of the actual data

- Date-Time Data Types - Redacted as random dates that are always different from those of the actual data

Prerequisites

- The user has to have **EXECUTE** privilege on the **DBMS_REDACT** package.
- Determine the **data type** of the column that will be redacted:
  - there are certain limitations for data types
  - Redacted column should not be used in Virtual Private Database (VPD) row filtering condition.
- Determine the **type of redaction** that should be performed: full, random, partial, regular expressions, or none.
- Determine conditions when the Data Redaction policy will be applied.
- Create redaction policy and subsequently add additionl columns by altering the policy.

Adding a Policy

```
BEGIN
  DBMS_REDACT.add_policy(
    object_schema => 'APP',
    object_name   => ,B_INFO',
    column_name   => 'PAN',
    policy_name   => 'APP_B_INFO_PAN',
    function_type => DBMS_REDACT.full,
    expression    => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =
                   ''JOZE_SENEGACNIK'''
  );
END;
/
```

Adding a Column to Existing Policy

```
BEGIN
  DBMS_REDACT.alter_policy (
    object_schema => 'APP',
    object_name   => 'B_INFO',
    policy_name   => 'APP_B_INFO_PAN',
    action        => DBMS_REDACT.add_column,
    column_name   => 'SECURE',
    function_type => DBMS_REDACT.full
  );
END;
/
```

Supported Policy Expressions

- The functionality is **limited**.

- Functions
  - SYS_CONTEXT()
  - V() and NV() in APEX
  - DOMINATES() in Label Security
- Operators:
  - Equivalency: =, !=
  - Comparison: >,<, <=, >=
  - Grouping: ()
  - Conjunction: AND, OR
  - Kexwords: IS, NOT, NULL

Example Conditions

- Expression parameter can be defined as follows:

```
expression => 'SYS_CONTEXT(
''USERENV'',''SESSION_USER'') = ''JOZE_SENEGACNIK'''

expression => 'SYS_CONTEXT(
''SYS_SESSION_ROLES'',''PAYMENT_INFO'') = ''TRUE'''

-- APEX Session States
expression => 'V(''APP_USER'') != ''larry@acme.com'' or
V(''APP_USER'') is null'
```

Data Redaction and Operations

- Users SYS and SYSTEM automatically have the EXEMPT REDACTION POLICY system privilege
- Data Redaction is also not enforced for users connected as SYSDBA.
- Running Data Pump one can get this error:

```
ORA-28081: Insufficient privileges - the command references a redacted object.
```

- The role DATAPUMP_EXP_FULL_DATABASE includes the EXEMPT REDACTION POLICY privilege which „disables‟ all defined policies.

- EXEMPT REDACTION POLICY system privilege disables data redaction if defined.

Insert Select / Create Table As Select

- In order to issue CTAS from a table protected by an active redaction policy, the user must have privileges to see the actual data on the source table.

- Or must have EXEMPT REDACTION POLICY privilege granted directly or through role.

Hacking By Inference

- Facts:
  - Data redaction for a column is used only when a column is in the selected list of columns.
  - Not used (can‘t really be used) on column values used in WHERE clause
- As long as one can execute SQL statement he can write a code to get the value by inference.
  - Credit Card Number (PAN) – certain format
  - One can use substring function in a loop defining one character at a time by values 0-9 and observing the result of SELECT statement.
  - Same can be done for other columns if it is known what kind of values they can have.
- Prevention:
  - Execution of SQL statements should be allowed only from PL/SQL code – no direct SQL statements allowed.

Thank you for your interest!

Q&A