
ORACLE®

Oracle Database 12c Release 2 CoreTech Seminar

12.2.0.1
Database Security

日本オラクル株式会社
クラウド・テクノロジー事業統括
Database & Exadataプロダクトマネジメント本部
西村克也
2016/10

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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Database Security

1. Transparent Data Encryption
2. Database Vault
3. Privilege Analysis
4. Data Redaction
5. DB Security Basic

Database Security

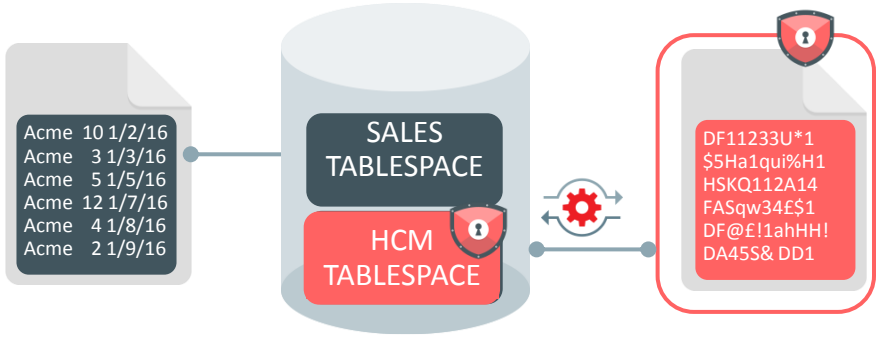
1. Transparent Data Encryption
2. Database Vault
3. Privilege Analysis
4. Data Redaction
5. DB Security Basic

Encryption Conversion

NEW IN 12.2

- SQLコマンドだけで表領域を暗号化・復号を実現
- 既存の表領域を暗号化する時間を大幅に短縮
- データにアクセス可能なまま表領域を暗号化
  - Online Encryption Conversion
- 表領域がオフライン時に暗号化
  - Offline Encryption Conversion
- 従来では暗号化できなかった、SYSTEM，SYSAUX，UNDO，TEMP表領域を暗号化させることが可能

  ※従来からUNDOとTEMP表領域は、TDE表領域暗号化が有効になっている場合、対象の暗号化表領域のデータ自体は暗号化されて格納されている
- Offline Encryption Conversionは、11.2.0.4 、12.1.0.2にバックポート



Online Encryption Conversion

- 12.2のみ使用
- 使用可能な暗号アルゴリズム は、AES(128,192,256bit), DES, ARIA, SEED, GOST
- 暗号化後に鍵の再作成やアルゴリズムの変更が可能。※OfflineでConversionされた表領域も可能
- SYSTEM, SYSAUX, UNDO表領域も暗号化可能
  ※ただし、SYSTEMとUNDOとユーザー表領域を同時にconversion処理を実行することは禁止
- 移行時は、一時的に対象の表領域と同等のディスク領域が必要
- Conversion処理実行中は、Data Pumpやトランスポータブル表領域は使用できない
- Conversion処理実行中は、以下のコマンドは実行できない
  - ADMINSTER KEY MANAGEMENT SET KEY
  - FLASHBACK DATABASE
- Online Encryption Conversion時は、REDOログは生成される
- 暗号化完了後に表領域にデータファイルを追加した場合は、追加されたファイルは暗号化される

Online Encryption Conversion

- 暗号化

```
ALTER TABLESPACE 表領域名 ENCRYPTION USING 暗号アルゴリズム ENCRYPT
FILE_NAME_CONVERT=(旧データファイル名, 新データファイル名);
※ 使用可能なアルゴリズム AES(128,192,256), DES(168), ARIA(128,192,256), SEED(128), GOST(256)
```

- 復号

```
ALTER TABLESPACE 表領域名 ENCRYPTION DECRYPT
FILE_NAME_CONVERTFILE_NAME_CONVERT=(旧データファイル名, 新データファイル名);
```

- 鍵再作成 or 暗号アルゴリズム変更

```
ALTER TABLESPACE 表領域名 ENCRYPTION USING 暗号アルゴリズム REKEY
FILE_NAME_CONVERT=(旧データファイル名, 新データファイル名);
```

※データファイルが複数の場合は、(旧DF名1,新DF名1,旧DF名2,新DF名2....)と記述する

Online Encryption Conversionの実行例

- 表領域作成

```
CREATE TABLESPACE TEST DATAFILE '/u01/app/oracle/oradata/ora001/test01.dbf' SIZE 100M ONLINE;
```

- 暗号化

```
ALTER TABLESPACE TEST ENCRYPTION USING 'AES192' ENCRYPT FILE_NAME_CONVERT=('test01.dbf',
'test01_enc.dbf');
```

- 実行結果の確認

```
SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;

 TS#       ENCRYPTIONALG  ENCRYPTED STATUS
---------- --------------------    -------------    ------------------------------
       7    AES192                YES              NORMAL

SELECT TABLESPACE_NAME,ENCRYPTED FROM DBA_TABLESPACES WHERE TABLESPACE_NAME='TEST';

TABLESPACE_NAME ENCRYPTED
-----------------------    ---------
TEST                       YES
```

Online Encryption Conversionの実行例

- 暗号アルゴリズムの変更

```
ALTER TABLESPACE TEST ENCRYPTION USING 'AES256' REKEY FILE_NAME_CONVERT=('test01_enc.dbf',
'test01_enc2.dbf');
```

- 実行結果の確認

```
SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;

 TS#       ENCRYPTIONALG  ENCRYPTED STATUS
---------- ---------------------    -------------    ------------------------------
       7   AES256                   YES              NORMAL
```

- 復号

```
ALTER TABLESPACE TEST ENCRYPTION DECRYPT FILE_NAME_CONVERT=('test01_enc2.dbf', 'test01.dbf');

SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;

レコードが選択されませんでした。
```

Online Encryption Conversionの再実行

- 移行中にディスク不足等の問題で処理が継続できなかった場合に処理を再実行させる

```
ALTER TABLESPACE 表領域名 ENCRYPTION ONLINE FINISH ENCRYPT / DECRYPT / REKEY
FILE_NAME_CONVERT=(旧データファイル名, 新データファイル名);
```

```
SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;
 TS#       ENCRYPTIONALG  ENCRYPTED STATUS
---------- --------------------    -------------    ------------------------------
       7    AES192                YES            ENCRYPTING       <----  NOMALではないので正常終了していない

- 再実行
 ALTER TABLESPACE TEST ENCRYPTION ONLINE FINISH ENCRYPT FILE_NAME_CONVERT = ('test01.dbf',
'test01_enc.dbf');

 TS#       ENCRYPTIONALG  ENCRYPTED STATUS
---------- --------------------    -------------    ------------------------------
       7    AES192                YES            NORMAL
```

Offline Encryption Conversion

- 12.2, 11.2.0.4, 12.1.0.2 で使用可能
- 暗号アルゴリズム は、AES128のみ
  - V$DATABASE_KEY_INFOで確認可能
- 暗号アルゴリズムの変更は、12.2 のみ
  - Online Encryption Conversionの機能を使用する
- SYSTEM, SYSAUX, UNDO表領域も暗号化可能 (12.2のみ)
- 移行のための特別なディスク領域は必要なし
- データファイル単位でのパラレル実行が可能
- 11.2.0.4, 12.1.0.2は、以下のパッチ適用が必要
  - Enable Transparent Data Encryption (TDE) Using Fast Offline Conversion in 11.2.0.4 and 12.1.0.2 （ドキュメントID 2148746.1）
  - 12.1は、12.1.0.2.160719 Database Proactive Bundle Patch (Jul 2016)に含まれる

Offline Encryption Conversion

- 暗号化

```
ALTER TABLESPACE TEST ENCRYPTION OFFLINE ENCRYPT;
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/ora001/test01.dbf' ENCRYPT;
```

- 復号

```
ALTER TABLESPACE TEST ENCRYPTION OFFLINE DECRYPT;
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/ora001/test01.dbf' DECRYPT;
```

- データファイルごとにパラレルに実行させた場合は、以下のコマンドをそれぞれを異なるセッションで実行する

```
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/ora001/test01.dbf' ENCRYPT;
ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/ora001/test02.dbf' ENCRYPT;
```

※11.2, 12.1はALTER DATABASE DATAFILEのコマンドのみ

Offline Encryption Conversionの実行例

- 表領域作成

```
CREATE TABLESPACE TEST DATAFILE '/u01/app/oracle/oradata/ora001/test01.dbf' SIZE 100M ONLINE;
```

- オフライン

```
ALTER TABLESPACE TEST OFFLINE;
```

- 暗号化

```
ALTER TABLESPACE TEST ENCRYPTION OFFLINE ENCRYPT;
```

- オンライン

```
ALTER TABLESPACE TEST ONLINE;
```

実行結果の確認

```
SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;

 TS#       ENCRYPTIONALG  ENCRYPTED STATUS
---------- ---------------------    -------------    ------------------------------
       7   AES128               YES            NORMAL
```

Offline Encryption Conversionの実行例

- オフライン

ALTER TABLESPACE TEST OFFLINE;

- 復号

ALTER TABLESPACE TEST ENCRYPTION OFFLINE DECRYPT;

- オンライン

ALTER TABLESPACE TEST ONLINE;

実行結果の確認

SELECT TS#, ENCRYPTIONALG,ENCRYPTEDTS,STATUS FROM V$ENCRYPTED_TABLESPACES;

レコードが選択されませんでした。

TEMP表領域の暗号化方法

- 12.2からCREATE TEMPORARY文でENCRYPTION ENCRYPTが指定可能

```
CREATE TEMPORARY TABLESPACE TEMP2 TEMPFILE '/u01/app/oracle/oradata/ora001/temp02.dbf'
SIZE 100M AUTOEXTEND ON ENCRYPTION ENCRYPT;
```

例）デフォルト一時表領域の変更

- 一時表領域の作成

```
CREATE TEMPORARY TABLESPACE TEMP2 TEMPFILE '/u01/app/oracle/oradata/ora001/temp02.dbf'
SIZE 100M AUTOEXTEND ON ENCRYPTION ENCRYPT;
```

- デフォルトの変更

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TEMP2;
```

- 既存の一時表領域の削除

```
DROP TABLESPACE TEMP;
```

データベース・フル暗号化の設定例

- DB mount起動

SQLPLUS / AS SYSDBA

STARTUP MOUNT

- マスター鍵オープン (※事前にマスタ鍵の設定が必要, Appendix参照)

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY “パスワード";

- 暗号化

ALTER TABLESPACE SYSTEM ENCRYPTION OFFLINE ENCRYPT;

ALTER TABLESPACE SYSAUX ENCRYPTION OFFLINE ENCRYPT;

ALTER TABLESPACE UNDOTBS1 ENCRYPTION OFFLINE ENCRYPT;

- DB起動

ALTER DATABASE OPEN;

- TEMP 表領域の暗号化設定

前ページ参照

Online vs. Offline Encryption Conversion

	Offline Conversion	Online Conversion
使用可能なバージョン	12.2 11.2.0.4, 12.1.0.2	12.2のみ
使用可能なアルゴリズム	AES128のみ	TDEで使用可能な すべてのアルゴリズム
いつ実行可能か？	Tablespaceがオフラインか Databaseがmount時	Tablespaceがオンラインか DatabaseがRead/Write Open時
追加のディスク領域が必要か？	必要なし	移行時は、一時的に対象の表領域と 同等のディスク領域が必要
DataGuardを使用している場合	PrimaryとStandbyを手動で conversionする必要がある	Primaryをconversionした後は、自動的 にStandby側はconversionされる
SYSTEM,SYSAUX,UNDO表領域の 暗号化	12.2のみ	12.2のみ
TEMP表領域は暗号化	作成は可能 (12.2のみ)	作成は可能 (12.2のみ)

Online vs. Offline Encryption Conversion

	Offline Conversion	Online Conversion
暗号化された表領域の復号	Offline Conversionで暗号化表領域は復号可能。ただし、UNDOを復号することは推奨しない	可能。ただしUNDOを復号することは推奨しない
表領域暗号鍵の再作成やローテーション	できない。ただし、12.2の場合は、Online ConversionのREKEYコマンドが使用可能	可能
パラレル実行	データファイルごとに複数のユーザーセッションで実行可能	表領域ごとに複数のユーザーセッションで実行可能
Conversionコマンドが途中で失敗した場合	暗号・復号を確実にするため再度コマンドを実行する	Finish句をつけたコマンドで再実行する

ENCRYPT_NEW_TABLESPACE初期化パラメータ

ENCRYPT_NEW_TABLESPACE = [CLOUD_ONLY] [ALWAYS] [DDL]

CLOUD_ONLY (デフォルト)

- Oracle Cloudの場合、CREATE TABLESPACEでENCRYPTION句を指定しなくても、AES128で表領域が自動的に暗号化される
- On-Premiseの場合、明示的に指定する必要がある。指定しなければ暗号化されない(従来通り)

ALWAYS

- CREATE TABLESPACEでENCRYPTION句を指定しなくても、AES128で表領域が自動的に暗号化される
- Oracle CloudとOn-Premiseもいずれも同じ動作

DDL

- CREATE TABLESPACEでENCRYPTION句を指定しなければ、表領域は暗号化されない(従来通り)
- Oracle CloudとOn-Premiseもいずれも同じ動作

FORCE KEYSTORE

- 自動ログイン・キーストアを使用している場合やキーストアがCLOSEしている場合でも、マスター暗号鍵のメンテナンス(鍵の作成、削除、再作成、Export/Import)等の操作が可能
- 自動ログイン・キーストアのMTA環境で、PDBの追加・Unplug/Plug等のオペレーションする場合、CDBのマスタ鍵をCLOSEする必要がないので、マスター鍵操作がDB全体に影響を与えない

- 鍵再作成

```
ADMINISTER KEY MANAGEMENT SET KEY USING TAG ‘20160922' FORCE KEYSTORE IDENTIFIED BY "oracle12c“
  WITH BACKUP;
```

- エクスポート

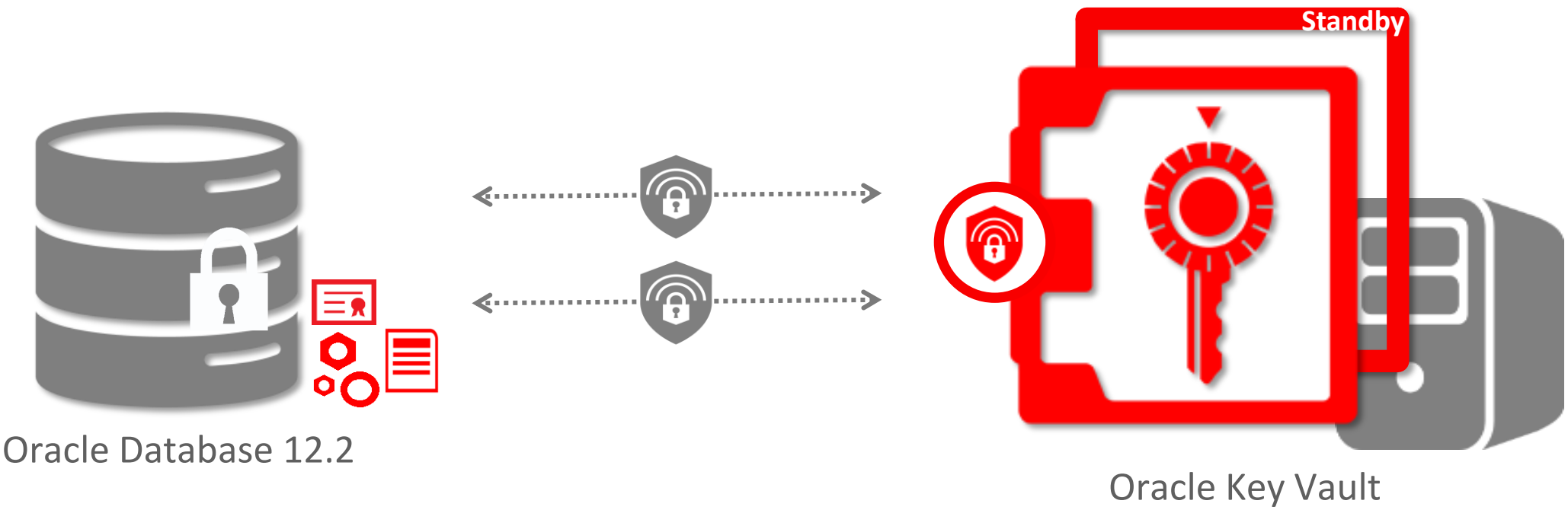
```
ADMINISTER KEY MANAGEMENT EXPORT KEYS WITH SECRET "my_secret" TO '/home/oracle/export.exp’
   FORCE KEYSTORE IDENTIFIED BY "oracle12c";
```

- インポート

```
ADMINISTER KEY MANAGEMENT IMPORT ENCRYPTION KEYS WITH SECRET "my_secret“
  FROM '/home/oracle/export.exp' FORCE KEYSTORE IDENTIFIED BY oracle12c WITH BACKUP;
```

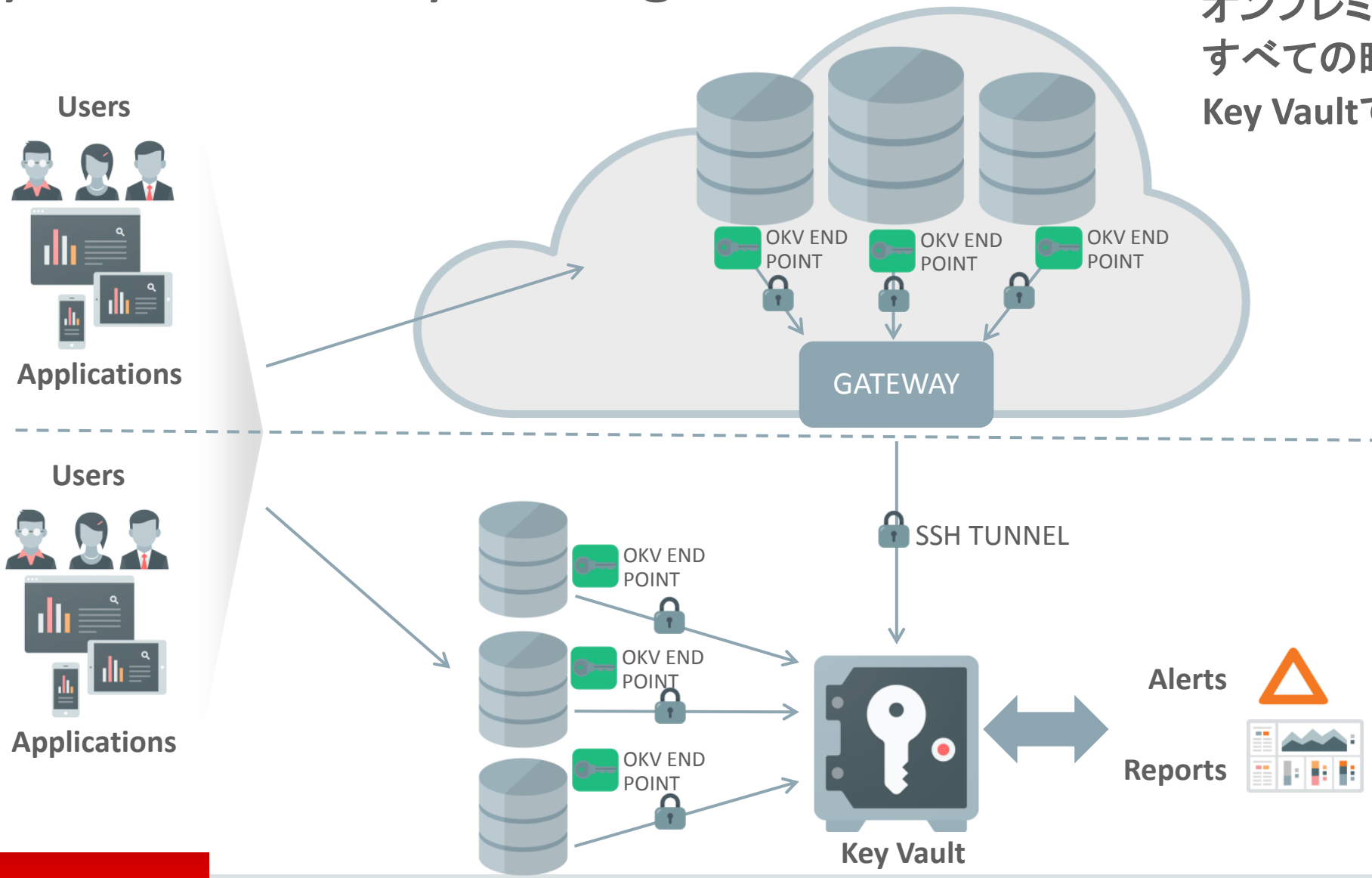
TDE + Oracle Key Vault

- Oracle Key Vaultは、暗号鍵をセキュアに保護するHSM (Hardware Security Module)
- TDEのマスタ暗号鍵をOKVに格納にし、セキュアに守りながら、使用状況の監視・不正使用のアラート
- DB – Key Vault間の通信が接続が切れると、暗号化表領域のアクセスはできなくなる
- Sqlnet.oraのENCRYPTION_WALLET_LOCATIONにOKV専用パラメータの追加(12.2)



Hybrid Cloud Key Management

オンプレミスとクラウドの
すべての暗号鍵を
Key Vaultでセキュアに一元管理



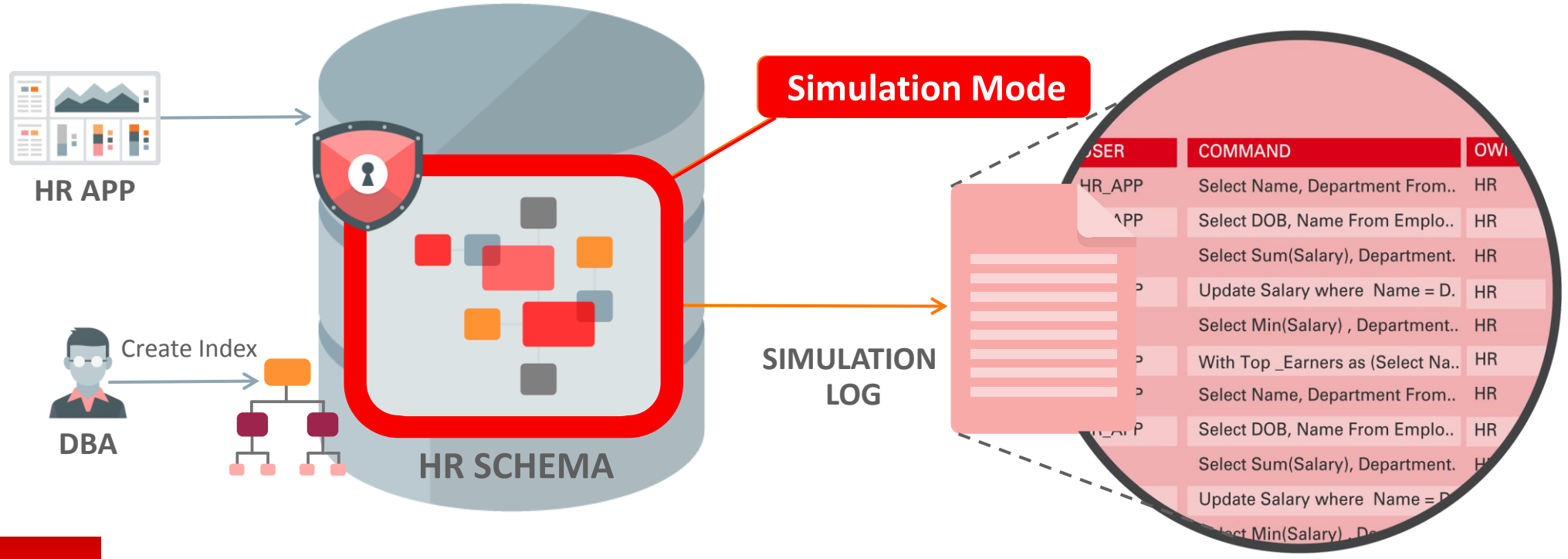
Database Security

1. Transparent Data Encryption
2. Database Vault
3. Privilege Analysis
4. Data Redaction
5. DB Security Basic

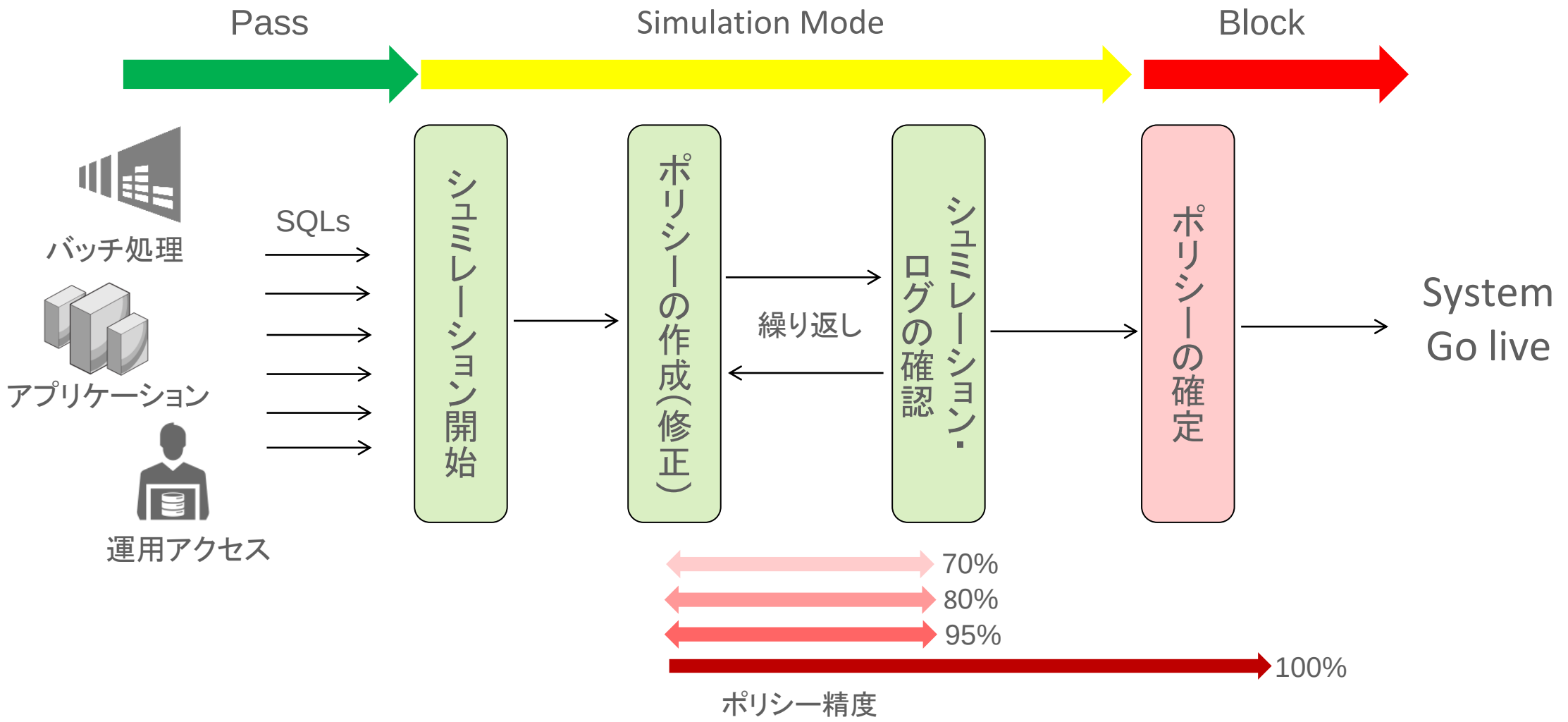
Simulation Mode

NEW IN 12.2

- Database Vault ポリシー (レルムやコマンドルール) をシュミレーションとして動作
  - ポリシー違反があった場合、アクセスをブロックするのではなく、シュミレーション・ログとして記録する
- シュミレーション・ログは、DBA_DV_SIMULATION_LOG ビューで参照可能
- 単体～システムテストのフェーズでポリシーの妥当性評価に最適



Simulation Modeを使った段階的なポリシー構築



Simulation Modeの設定

- ポリシー、レルム、コマンドルール作成時に、enabledの項目にDBMS_MACUTL.G_SIMULATIONを指定する
  - ポリシーをブロックに変更する場合は、DBMS_MACUTL.G_YESを指定
- DBA_DV_SIMULATION_LOGが参照するシュミレーションログの実表は、DVSYS.SIMULATION_LOG$
  - 削除する場合は、DV_OWNER でDELETE文を実行する

レルム

```
BEGIN
DBMS_MACADM.CREATE_REALM(
  realm_name    => 'EMPLOYEES_realm‘,
  enabled          => DBMS_MACUTL.G_SIMULATION,
  audit_options  => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type      => 1);
END;
/
```

コマンドルール

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command        => 'SELECT',
  rule_set_name   => 'Disabled',
  object_owner    => 'HR',
  object_name     => 'JOBS',
  enabled    =>    DBMS_MACUTL.G_SIMULATION);
END;
/
```

Simulation Modeの実行例

アクセス制御
①SYSユーザーはHRのEMPLOYEES表にアクセスできない
②HRユーザーはアクセス可能

- シュミレーションモードでレルム作成

```
BEGIN
DBMS_MACADM.CREATE_REALM(
  realm_name    => 'EMPLOYEES_realm',
  description      => 'Realm to protect HR.EMPLOYEES',
  enabled          => DBMS_MACUTL.G_SIMULATION,
  audit_options  =>
DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type      => 1);
END;
/
```

- EMPLOYEES表をレルムに追加

```
BEGIN
DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'EMPLOYEES_realm',
  object_owner => 'HR',
  object_name  => 'EMPLOYEES',
  object_type  => 'TABLE');
END;
/
```

- HRをレルム・オーナーに指定

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'EMPLOYEES_realm',
  grantee      => 'HR',
  auth_options =>
DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

Simulation Modeの実行例

SYSユーザー
sqlplus / as sysdba
select employee_id,email,salary from hr.employees;

```
EMPLOYEE_ID  EMAIL          SALARY    <---------- アクセス可能
-----------  -------------  ----------
        100  SKING          24000
        101  NKOCHHAR       17000
```

HRユーザー
con hr/hr
select employee_id,email,salary from hr.employees;

```
EMPLOYEE_ID  EMAIL          SALARY    <---------- アクセス可能
-----------  -------------  ----------
        100  SKING          24000
        101  NKOCHHAR       17000
```

DBA_DV_SIMULATION_LOGビューでログ確認

- SYSからのEMPLOYEES表へのアクセスは、本来はレルム違反(アクセスできない)ので、シュミレーション・ログに記録される

```
SELECT TO_CHAR(TIMESTAMP,'yyyy/mm/dd hh24:mi:ss') TIMESTAMP, USERNAME, COMMAND,
VIOLATION_TYPE, SQLTEXT from DBA_DV_SIMULATION_LOG;

TIMESTAMP           USER  COMMAND  VIOLATION_       SQLTEXT
------------------- ----- -------- ---------------- ------------------------------------------------
2016/06/09 01:15:15 SYS   SELECT   Realm Violation   SELECT EMPLOYEE_ID,EMAIL,SALARY FROM
                                                     HR.EMPLOYEES

2016/06/09 01:17:55 SYS   SELECT   Realm Violation   SELECT EMPLOYEE_ID,EMAIL,SALARY FROM
                                                     HR.EMPLOYEES

2016/06/08 16:56:58 SYS   SELECT   Realm Violation   SELECT * FROM HR.EMPLOYEES
```

レルムをSimulationからEnableに変更

```
- レルムをEnableに変更
BEGIN
DVSYS.DBMS_MACADM.UPDATE_REALM(
  realm_name    => 'EMPLOYEES_realm',
  description      => 'Realm to protect HR.EMPLOYEES',
  enabled          => DBMS_MACUTL.G_YES);
END;
/
```

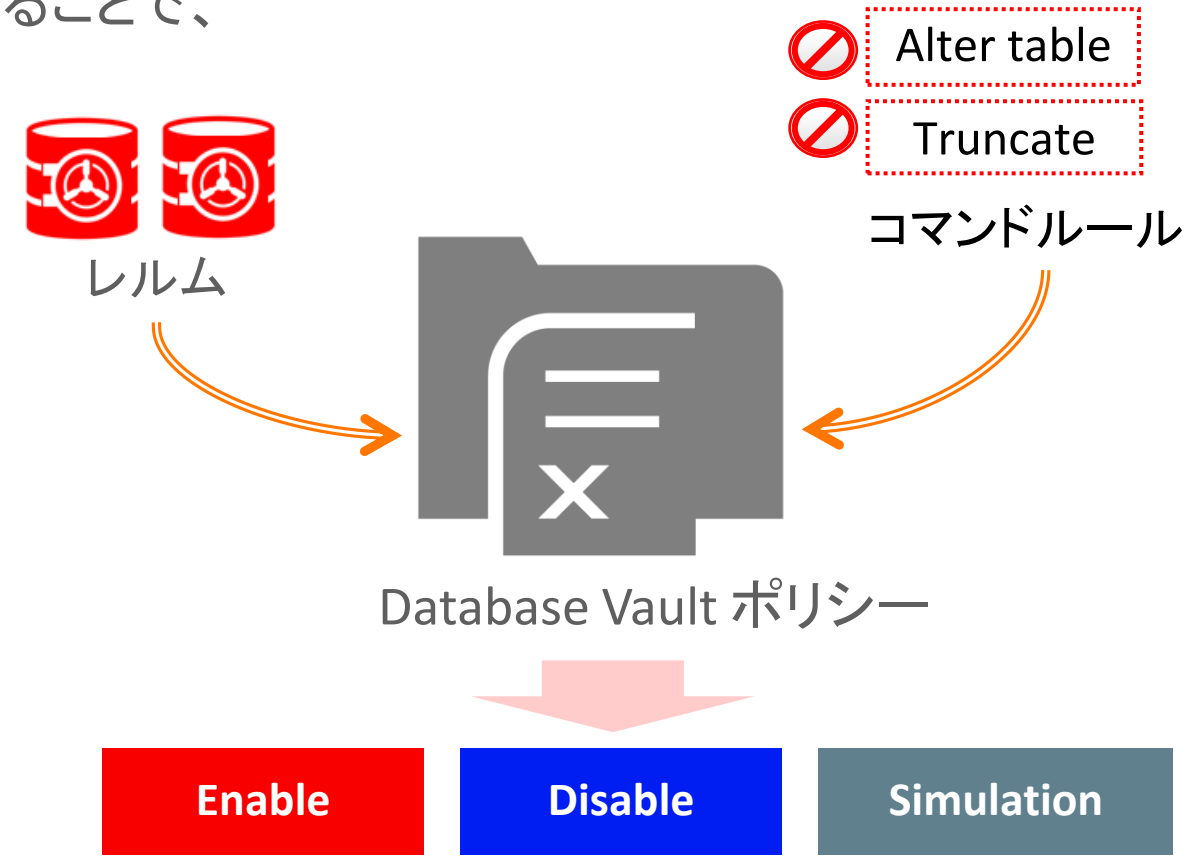
```
sqlplus / as sysdba
select employee_id,email,salary from hr.employees;

行1でエラーが発生しました。:                    <---------- SYSはアクセス不可
ORA-01031: 権限が不足しています

con hr/hr
select employee_id,email,salary from hr.employees;

EMPLOYEE_ID  EMAIL          SALARY    <---------- HRはアクセス可能
-----------  -------------  ----------
        100  SKING          24000
```

Database Vault ポリシーによる一括制御

NEW IN 12.2

- レルムやコマンドルールそれぞれで有効・無効化ではなく、
  12.2からの新しい定義であるDatabase Vaultのポリシーに必要なものをレルム等を含め、
  ポリシー単位での有効・無効・シュミレーションの制御が可能
- アプリケーションやセキュリティの単位でまとめることで、
  ポリシーの運用・可視性の向上

```
BEGIN
  DVSYS.DBMS_MACADM.CREATE_POLICY(
    policy_name => 'HR Applicaton',
    description =>    'All Object for HR APP',
    policy_state =>   DBMS_MACUTL.G_ENABLE);
END;
```



MTA環境の共通レルム・コマンドルール

NEW IN 12.2

- すべてのPDBで共通のポリシー制御を可能にする
- MTAのアプリケーションルートに共通レルムとして作成
- CREATE_REALM，CREATE_COMMAND_RULEに追加されたscopeパラメータ
  - DBMS_MACUTL.G_SCOPE_LOCAL
  - DBMS_MACUTL.G_SCOPE_COMMON

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
   realm_name     => 'Performance Statistics Realm',
   description    => 'Realm to measure performance',
   enabled        => DBMS_MACUTL.G_YES,
   audit_options  => DBMS_MACUTL.G_REALM_AUDIT_FAIL + DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
   realm_type     => 1,
   realm_scope    => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

ORACLE

Database Security

1. Transparent Data Encryption
2. Database Vault
3. **Privilege Analysis**
4. Data Redaction
5. DB Security Basic

Privilege Analysis新機能

NEW IN 12.2

- PL/SQLのコンパイル時の権限
  - PL/SQLプロシージャがコンパイル時に、ORA$DEPENDENCYというキャプチャ名でPL/SQL内で依存している権限の記録、権限を記録する (12.1)
  - PL/SQLプロシージャを実行に必要な権限が、ユーザーからREVOKEされた等の権限変更があった場合、その変更を検知し、使用・未使用を記録できる(12.2)
- コードベース・アクセス制御の権限
  - PL/SQL実行者がコートベース・アクセス制御でアクセスした場合の権限を記録する(12.1)
  - PL/SQL実行者がコートベース・アクセス制御でアクセスした場合の権限がどこから付与されたものか、その詳細まで記録する(12.2)
- 権限キャプチャの実行
  - 定義したキャプチャ・ポリシーを同時に複数は実行できない(12.1)
  - 定義したキャプチャ・ポリシーを同時に複数は実行し、その結果レポートを見比べて確認できる(12.2)
- DBA_UNUSED_GRANTSビューの追加
  - ユーザーやロールに付与された権限で使用されていないものをレポート

例) コードベース・アクセス制御

1.Privilege Analysisのキャプチャ作成・開始 (対象はHRとFIN)

```
sqlplus / as sysdba;

BEGIN
DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
 name            => 'HR_Access',
 type            =>  DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
 condition       => ‘SYS_CONTEXT(’‘USERENV’‘, ’‘SESSION_USER’‘)
                    IN ''HR'', ''FIN'')');
END;
/

BEGIN
 DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE (
   name         => 'HR_Access',
   run_name  => 'HR_Access_001');
END;
/
```

2.コードベース・アクセス制御でのアクセスの実行

```
GRANT CONNECT TO FIN IDENTIFIED BY fin;
GRANT CREATE ROLE TO HR;

conn hr/hr
```

例) コードベース・アクセス制御

print_employeesのプロシージャを作成
(HRのemployeeとdepartment表をアクセスする)

```
create or replace procedure print_employees
authid current_user
as
begin
 dbms_output.put_line(rpad('ID', 10) ||
                rpad('First Name', 15)  ||
                rpad('Last Name', 15)   ||
                rpad('Email', 15)       ||
                rpad('Phone Number', 20));
 for rec in (select e.employee_id, e.first_name, e.last_name,
                e.email, e.phone_number
            from hr.employees e, hr.departments d
           where e.department_id = d.department_id
            and d.department_name =
               sys_context('userenv', 'current_user'))
 loop
  dbms_output.put_line(rpad(rec.employee_ID, 10)  ||
                rpad(rec.first_name, 15)   ||
                rpad(rec.last_name, 15)    ||
                rpad(rec.email, 15)        ||
                rpad(rec.phone_number, 20));
 end loop;
end;
/
```

hr_clerkロールを作成
(print_employeesへのexecute権限を含め、finに付与)

```
CREATE ROLE hr_clerk;
GRANT EXECUTE ON print_employees TO hr_clerk;
GRANT hr_clerk TO fin;
```

finでprint_employeesを実行する

```
conn fin/fin
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM
HR.EMPLOYEES;
EXEC HR.print_employees;
```

エラーになる。なぜか？
finはhrのEMPLOYEES表のSELECT権限を持たないから

hrでview_emp_roleにEMPLOYEES表とDEPARTMENT表へのselectを含め、そのロールをprint_employeesプロシージャに与える

```
conn hr/hr
CREATE ROLE view_emp_role;
GRANT SELECT ON HR.EMPLOYEES TO view_emp_role;
GRANT SELECT ON HR.DEPARTMENTS TO view_emp_role;
GRANT view_emp_role TO PROCEDURE HR.print_employees;
```

例) コードベース・アクセス制御

再度、SQLを実行

```
conn fin/fin
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM
HR.EMPLOYEES;
EXEC HR.print_employees;
```

EMPLOYEES表への直接アクセスできないが、プロシージャからのアクセスは実行できた。なぜか？
print_employeesプロシージャ自身にview_emp_roleロールが付与されたから <----コードベース・アクセス制御

3.Privilege Analysisの停止・レポート

```
conn / as sysdba;
EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('HR_Access');

BEGIN
 DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT (
   name        => 'HR_Access',
   run_name  => 'HR_Access_001');
END;
/
```

```
set line 200
col capture format a10
col username format a10
col obj_priv format a13
col object_owner format a10
col object_name format a28
col path format a50

select capture, username,  obj_priv, object_owner, object_name, path
from DBA_USED_PRIVS where run_name = 'HR_ACCESS_001' and
USERNAME='FIN';

USERNAME OBJ_PRIV  OBJECT_OWN OBJECT_NAME  PATH
-------- --------- ---------- -----------  --------------------------------------------------
FIN      SELECT    HR         EMPLOYEES
GRANT_PATH('HR.PRINT_EMPLOYEES', 'VIEW_EMP_ROLE')
FIN      SELECT    HR         DEPARTMENTS
GRANT_PATH('HR.PRINT_EMPLOYEES', 'VIEW_EMP_ROLE')
```

FINが、HRのEMPLOYEES とDEPARTMENTS にSELECT権限を使用した
その権限パスは、HRの‘HR.PRINT_EMPLOYEES パッケージ
の’VIEW_EMP_ROLEロールからを意味する

Database Security

1. Transparent Data Encryption
2. Database Vault
3. Privilege Analysis
4. **Data Redaction**
5. DB Security Basic

Data Redaction新機能

NEW IN 12.2

- リダクションポリシー
  - 表やビューに設定できるリダクション・ポリシーの条件は全体で一つ(12.1)
  - 表やビューのそれぞれの列ごとにリダクション・ポリシーの条件が指定可能(12.2)
- リダクションポリシー条件のライブラリ化
- リダクション結果のNULL値対応
- CLOB、NCLOBの正規表現 リダクション対応
- リダクション条件での使用可能文字の拡張
  - =, !=, >, <, >=, <= (12.1)
  - AND, OR, IN, NOT IN, =, !=, <>, <, >, >=, <= (12.2)

ORACLE

列ごとのリダクション・ポリシー

- それぞれのユーザーごとにリダクションする列を変更する
  - 管理者(ADMIN) : すべての列が参照可能
  - アプリケーション(APP) : SALARY列だけをリダクション
  - 開発者(DEV) ：FIST_NAME, LAST_NAME, SALARY列をリダクション

ADMIN

Conn ADMIN/ADMIN

FIRST_NAME	LAST_NAME	PHONE_NUMBER	SALARY
Steven	King	515.123.4567	24000
Neena	Kochhar	515.123.4568	17000
Lex	De Haan	515.123.4569	17000
Alexander	Hunold	590.423.4567	9000
Bruce	Ernst	590.423.4568	6000
David	Austin	590.423.4569	4800

APP

Conn APP/APP

FIRST_NAME	LAST_NAME	PHONE_NUMBER	SALARY
Steven	King	515.123.4567	0
Neena	Kochhar	515.123.4568	0
Lex	De Haan	515.123.4569	0
Alexander	Hunold	590.423.4567	0
Bruce	Ernst	590.423.4568	0
David	Austin	590.423.4569	0

DEV

Conn DEV/DEV

FIRST_NAME	LAST_NAME	PHONE_NUMBER	SALARY
S{h^%s	¥E88	515.123.4567	0
*SW[W	'c`HTfP	515.123.4568	0
lep	bjX:*']	515.123.4569	0
Y{}-8>GDa	]¥B??,	590.423.4567	0
DrUN8	)IGIu	590.423.4568	0
G/SyP	“k uV{	590.423.4569	0

列ごとのリダクション・ポリシー　設定例 1/3

1. リダクションポリシーの作成

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema =>'HR',
   object_name    =>'EMPLOYEES',
   policy_name    =>'EMPLOYEE_POLICY',
   expression    => 'sys_context(''userenv'',''session_user'') NOT IN (''ADMIN'') or
                        sys_context(''userenv'',''session_user'') is null',
   column_name    =>'salary',
   function_type=> DBMS_REDACT.FULL);
END;
/

BEGIN
 DBMS_REDACT.ALTER_POLICY (
   object_schema    => 'HR ',
   object_name      => 'EMPLOYEES',
   policy_name      => 'EMPLOYEE_POLICY',
   action               =>  DBMS_REDACT.ADD_COLUMN,
   column_name     =>  'FIRST_NAME',
   function_type    =>  DBMS_REDACT.RANDOM);
END;
/
```

デフォルトポリシーは
ADMIN以外

```
BEGIN
 DBMS_REDACT.ALTER_POLICY (
   object_schema    =>' HR ',
   object_name      =>' EMPLOYEES',
   policy_name      => 'EMPLOYEE_POLICY',
   action               =>  DBMS_REDACT.ADD_COLUMN,
   column_name     =>  'LAST_NAME',
   function_type    =>  DBMS_REDACT.RANDOM);
END;
/

BEGIN
 DBMS_REDACT.ALTER_POLICY (
   object_schema   => 'HR ',
   object_name     =>' EMPLOYEES',
   policy_name     =>' EMPLOYEE_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name    =>  'PHONE_NUMBER',
   function_type    =>  DBMS_REDACT.PARTIAL,
   function_parameters  => 'VVVFVVVFVVVV,VVV.VVV.VVVV,*,1,6');
END;
/
```

※ここまでのポリシーは、ADMINユーザー以外はFIRST_NAME, LAST_NAME, PHONE_NUMBER列がリダクションされる

列ごとのリダクション・ポリシー　設定例 2/3

- リダクションポリシー条件のライブラリ作成

2. リダクションポリシーの条件の作成

ポリシー条件に名前をつけて
ライブラリ化

```
BEGIN
 DBMS_REDACT.CREATE_POLICY_EXPRESSION(
   POLICY_EXPRESSION_NAME    => 'Admin_APP',
   expression     => 'sys_context(''userenv'',''session_user'') NOT IN (''ADMIN'',''APP'') or sys_context(''userenv'',''session_user'') is null');
END;
/
```

ADMIN, APPユーザー以外

```
BEGIN
 DBMS_REDACT.CREATE_POLICY_EXPRESSION(
   POLICY_EXPRESSION_NAME    => 'Admin_APP_Dev',
   expression     => 'sys_context(''userenv'',''session_user'') NOT IN (''ADMIN'',''APP'',''DEV'') or sys_context(''userenv'',''session_user'') is
null');
END;
/
```

ADMIN, APP, DEVユーザー以外

列ごとのリダクション・ポリシー　設定例 3/3

3. 列ごとにポリシーを変更

```
BEGIN
DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
  object_schema        => 'HR',
  object_name          => 'EMPLOYEES',
  column_name          => 'FIRST_NAME',
  policy_expression_name => 'Admin_APP');
END;
/

BEGIN
DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
  object_schema        => 'HR',
  object_name          => 'EMPLOYEES',
  column_name          => 'LAST_NAME',
  policy_expression_name => 'Admin_APP');
END;
/

BEGIN
DBMS_REDACT.APPLY_POLICY_EXPR_TO_COL(
  object_schema        => 'HR',
  object_name          => 'EMPLOYEES',
  column_name          => 'PHONE_NUMBER',
  policy_expression_name => 'Admin_APP_Dev');
END;
/
```

リダクション列の条件を作成した
ライブラリを指定して変更

FIRST_NAME列の条件を
Admin_APPポリシーに変更

LAST_NAME列の条件を
Admin_APPポリシーに変更

PHONE_NUMBER列の条件を
Admin_APP_DEVポリシーに変更

Database Security

1. Transparent Data Encryption
2. Database Vault
3. Privilege Analysis
4. Data Redaction
5. DB Security Basic

SYSRAC管理者権限

NEW IN 12.2

- RACのクラスタウェアを制御するための最小限の権限
  - 接続例) CONNECT / AS SYSRAC
- 以下のユーティリティの操作が可能
  - CRSCTL
  - SRVCTL
  - OCRCONFIG
  - CLUVFY
- 従来のSYSDBAによるクラスタウェアの操作リスクを減らす
- パスワードファイル、ネットワーク認証は利用できない
- OS認証でのグループ名
  - racdba(Linux/Unix)
  - ORA_HOMENAME_SYSRAC(Windows)

INACTIVE ACCOUNT TIME

NEW IN 12.2

- データベース・ユーザーが一定期間ログオンされない状態が続いた場合、自動的にロックする
- デフォルトのプロファイルは、UNLIMITED

```
PROFILE         RESOURCE_NAME                    RESOURCE_TYPE      LIMIT
--------------- -------------------------------- ------------------ ---------------
DEFAULT         FAILED_LOGIN_ATTEMPTS            PASSWORD           10
DEFAULT         INACTIVE_ACCOUNT_TIME            PASSWORD           UNLIMITED
DEFAULT         PASSWORD_GRACE_TIME              PASSWORD           7
DEFAULT         PASSWORD_LIFE_TIME               PASSWORD           180
DEFAULT         PASSWORD_LOCK_TIME               PASSWORD           1
DEFAULT         PASSWORD_REUSE_MAX               PASSWORD           UNLIMITED
DEFAULT         PASSWORD_REUSE_TIME              PASSWORD           UNLIMITED
DEFAULT         PASSWORD_VERIFY_FUNCTION         PASSWORD           NULL

-> 30日に変更
ALTER PROFILE DEFAULT LIMIT INACTIVE_ACCOUNT_TIME 30;
PROFILE         RESOURCE_NAME                    RESOURCE_TYPE      LIMIT
--------------- -------------------------------- ------------------ ---------------
DEFAULT         INACTIVE_ACCOUNT_TIME            PASSWORD           30
```

管理者アカウントの認証強化

NEW IN 12.2

- 管理者権限 (SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDB, SYSKM)を持つ管理者アカウントに以下のパスワード・プロファイルの制限を強制できる
  - FAILED_LOGIN_COUNT　(ログイン失敗許容回数)
  - PASSWORD_LOCK_TIME　(ログイン失敗ロック期間)
  - PASSWORD_GRACE_TIME (パスワード期限切れ猶予期間)
  - PASSWORD_LIFE_TIME　(パスワード有効期限)
- 管理者アカウントでもLOCK、EXPIREDされる
- 上記の制御を有効化するには、パスワードファイルが12.2の必要がある

※パスワードファイルのマイグレーション手順

パスワードファイルのバージョンの確認
orapwd describe file=orapwora001

パスワード・ファイルの12.2へのマイグレーション例
mv orapwora001 orapwora001_bk
orapwd file=orapwora001 input_file=orapwora001_bk format=12.2

※パスワードファイルの場所
$ORACLE_HOME/dbs

設定確認例

1. パスワードファイルが12.2にマイグレーション完了済であること
2. テスト用プロファイルの作成

   CREATE PROFILE test_profile LIMIT

   FAILED_LOGIN_ATTEMPTS 2

   PASSWORD_LOCK_TIME 30;
3. テスト用ユーザーの作成

   CREATE USER TEST IDENTIFIED BY TEST;

   GRANT SYSDBA TO TEST;
4. ユーザーのプロファイルを変更

   ALTER USER TEST PROFILE test_profile;
5. パスワードファイルにプロファイルが設定されたのを確認

   USERNAME ACCOUNT_STATUS PASSWORD_PROFILE

   ---------- ---------- ----------

   TEST OPEN TEST_PROFILE

6. 念のためOS認証を無効にするために、SQLNET.ORAに以下を追記する

   SQLNET.AUTHENTICATION_SERVICES=(none)
7. ログインの失敗を繰り返す

   sqlplus test/testa as sysdba

   ERROR:

   ORA-28000: アカウントがロックされています。

   ※OS認証時にはこの制御は効かない

ロール・ベースの監査設定

NEW IN 12.2

AUDIT POLICY ポリシー名 BY USERS WITH GRANTED ROLES ロール名1,ロール名2

- 12.1では、AUDIT POLICY ポリシー名 [BY] [EXCEPT] ユーザー名、のように全・個別のユーザーに対して監査設定を行う
- 12.2からは、ロール自身に対して監査の設定を行うことが可能
- そのロールを付与されているユーザーは、自動的に監査される
  - 例えば、DBAロール自体に監査設定をした場合、DBAを付与されているユーザーがDBAロールに含まれる権限を使用したデータベース操作をした場合、UNIFIED AUDITに記録される
  - 監査対象の主体がロールため、多くのDBユーザー個別に監査設定を考える必要はなくなる
  - 新規ユーザーに対しても、監査設定を忘れずに適用できる

ロール・ベースの監査設定例

- DBAロールに対して監査

```
CREATE AUDIT POLICY DBA_ROLE_AUDIT ROLES DBA;
AUDIT POLICY AUDIT POLICY DBA_ROLE_AUDIT  BY USERS WITH GRANTED ROLES DBA;

SELECT DBUSERNAME,SQL_TEXT,SYSTEM_PRIVILEGE_USED,UNIFIED_AUDIT_POLICIES FROM
UNIFIED_AUDIT_TRAIL WHERE UNIFIED_AUDIT_POLICIES ='DBA_ROLE_AUDIT‘;
```

DBUSERNAME	SQL_TEXT	SYSTEM_PRIVILEGE_USED	UNIFIED_AUDIT_POLICIES
SCOTT		CREATE SESSION	DBA_ROLE_AUDIT
SCOTT	select * from hr.employees	SELECT ANY TABLE	DBA_ROLE_AUDIT
SCOTT	insert into hr.jobs values('4000','DUMMY',9000,12000)	INSERT ANY TABLE	DBA_ROLE_AUDIT
SCOTT	delete hr.jobs where JOB_TITLE='dummy'	DELETE ANY TABLE	DBA_ROLE_AUDIT

-->SCOTTユーザーがDBAロールにあるANY権限を使って、HRの表にアクセスしたことを監査

Auditの変更点

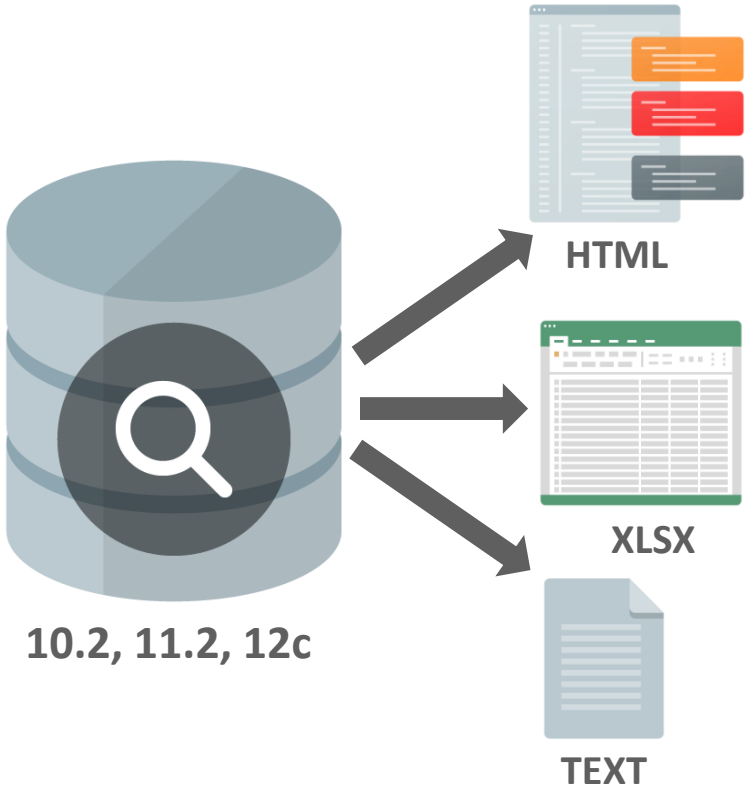
NEW IN 12.2

- SELECT ANY TABLE権限でAUDSYSのスキーマへのアクセスは禁止
- SELECT ANY DICTIONARY権限のみがAUDSYSのスキーマへのアクセス可能
- AUDSYSの内部表がUnified AuditのREADパフォーマンス向上のために改良されているため、12.1からのアップグレード後は、DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDSで監査データの移行が必要
- Virtual Private Database(VPD)が実行された場合のwhere句がログに記録される

Database Security Assessment Tool (DBSAT)

Oracleが提供するセキュリティ診断ツール

- データベースをスキャンし、DBのセキュリティを網羅的にチェックするツール
  - 基本構成、ユーザーアカウント、権限管理、暗号化、監査・・・
- Pythonで作成された完全自動化スクリプト
- HTML, XLSX, TEXTの形式でそれぞれレポート出力
- 出力レポートは英語　※日本オラクルから日本語化パッチ提供
- ダウンロード先
  - Doc ID 2138254.1
    Oracle Database Security Assessment Tool (DBSAT)
- ドキュメント
  - Oracle Database Security Assessment Tool User Guide
    http://docs.oracle.com/cd/E76178_01/index.htm



リファレンス
マニュアル・ドキュメント

- Database Security Guide
  http://docs.oracle.com/cd/E82638_01/DBSEG/toc.htm
- Database Advanced Security Guide
  http://docs.oracle.com/cd/E82638_01/ASOAG/toc.htm
- Database Vault Administrator's Guide
  http://docs.oracle.com/cd/E82638_01/DVADM/toc.htm

Safe Harbor Statement

The preceding 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®