Oracle® Database

Learning Key 20c New Features for Database Administrators

F25017-04
May 2020

Oracle Database Learning Key 20c New Features for Database Administrators,

F25017-04

Primary Author: Dominique Jeunot

Contributors: Alan Williams, Saurabh Naresh Netravalkar, Vijayendra Lakkundi, Jim Stenoish, Bill Beauregard, Gregg Christman, , Andy Rivenes, Teck Hua Lee, Vijayendra Lakkundi, Nigel Bayliss, Allison Holloway, Sathya Jaganathan, Huagang Li, Kelly Smith, Mark Scardina, Bill Burton, Ravi Thammaiah, Hermann Baer, Christopher Jones, Abhishek Munnolimath, Shashaanka Agrawal, Dominic Giles, Daniel Overby Hansen, Srikanth Bellamkonda, Daniel Overby Hansen, Krishna Mohan, Preetam Ramakrishna

Contents

1 Learning Key 20c New Features for Database Administrators

Preface

This document describes new features implemented in Oracle Database 20c.

Audience

Read the "*Oracle Database Learning Key 20c New Features for Database Administrators*" if you want to learn about features, options, and enhancements that are new in Oracle Database 20c and benefit from practices to better understand the features use.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Database 20c documentation set:

- *Oracle Database New Features*
- *Oracle Database Error Messages*
- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Learning Key 20c New Features for Database Administrators

The learning guide shows the major new features and enhancements introduced in Oracle Database 20c, for which you can get more details and even practices to experiment them.

For other new features that are not providing any more details or practices, refer to the Oracle® Database New Features Guide 20c.

- Practices Environment
  If you plan to test the practices available in different sections, the practices are designed to be independent from one another. This is the reason why, in case a particular configuration is already enabled in your testing database, the recreation of your testing database is suggested, but not mandatory.
- Security Solutions
- Performance and High-Availability Options
- Tools and Languages
- Upgrades, Patching and Migrations

Practices Environment

If you plan to test the practices available in different sections, the practices are designed to be independent from one another. This is the reason why, in case a particular configuration is already enabled in your testing database, the recreation of your testing database is suggested, but not mandatory.

- Practices Environment on Oracle Database Cloud Preview

Practices Environment on Oracle Database Cloud Preview

1. Create an instance of a 20c Cloud Preview Database running in Oracle Cloud Infrastructure following the instructions explained in the *Create an Oracle Cloud Infrastructure VM Database* tutorial. Name the CDB `CDB20` and its pluggable database `PDB20`.
2. Once your 20c Cloud Preview `CDB20` and `PDB20` are created, an alias entry is automatically created in `/u01/app/oracle/homes/OraDB20Home1/network/admin/tnsnames.ora`. It is recommended to add an alias entry in `/u01/app/oracle/homes/OraDB20Home1/network/admin/tnsnames.ora` for `PDB20` to provide an easier connection to `PDB20`.

   ```
   $ cat /u01/app/oracle/homes/OraDB20Home1/network/admin/tnsnames.ora
   LISTENER_CDB20=(ADDRESS=(PROTOCOL=TCP)(HOST=host_value)(PORT=1521))

   CDB20_iad1bw=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_value)
   ```

```
(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=CDB20_iad1bw.subnetname.dbvcn.oraclevcn.com)))
$
```

3. Create an alias entry by copying the CDB alias entry, replace the CDB alias name with your PDB name, and the CDB service name with your PDB service name.

```
$ vi /u01/app/oracle/homes/OraDB20Home1/network/admin/tnsnames.ora
LISTENER_CDB20=(ADDRESS=(PROTOCOL=TCP)(HOST=host_value)(PORT=1521))

CDB20_iad1bw=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_value)
(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=CDB20_iad1bw.subnetname.dbvcn.oraclevcn.com)))

PDB20=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_value)(PORT=1521))
(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=PDB20.subnetname.dbvcn.oraclevcn.com)))
$
```

Do the same operation for each new PDB created in the CDB.

4. Test the connection to CDB20.

```
$ sqlplus sys@CDB20_iad1bw AS SYSDBA

SQL*Plus: Release 20.0.0.0.0 - Production on Thu Apr 2 15:20:34 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password_defined_during_DBSystem_creation

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SHOW CON_NAME

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

5. Test the connection to PDB20.

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password_defined_during_DBSystem_creation
Connected.
SQL> SHOW CON_NAME

CON_NAME
------------------------------
PDB20
```

```
SQL> EXIT
$
```

6. Download the `Cloud_Preview_20c_labs.zip` zip file from Oracle Web Content in the directory `/home/oracle` on your VM and unzip the file.

```
$ cd /home/oracle
$ unzip Cloud_Preview_20c_labs.zip
Archive:  Cloud_Preview_20c_labs.zip
inflating: labs/update_pass.sh
   creating: labs/M104785GC10/
  inflating: labs/M104785GC10/hr_cre.sql
  inflating: labs/M104785GC10/hr_idx.sql
  inflating: labs/M104785GC10/hr_main_new.sql
  inflating: labs/M104785GC10/hr_code.sql
  inflating: labs/M104785GC10/hr_main.sql
  inflating: labs/M104785GC10/hr_drop_new.sql
  inflating: labs/M104785GC10/hr_analz.sql
  inflating: labs/M104785GC10/hr_drop.sql
  inflating: labs/M104785GC10/profile.sql
  inflating: labs/M104785GC10/flashback.sql
...
  inflating: labs/M104782GC10/create_PDB20.sql
  inflating: labs/M104782GC10/tnsnames.ora
  inflating: labs/M104782GC10/hr_comnt.sql
  inflating: labs/M104782GC10/hr_popul.sql
  inflating: labs/M104782GC10/create_CDB20.sh
  inflating: labs/M104782GC10/listener.ora
$
```

7. Launch the `/home/oracle/labs/update_pass.sh` shell script. The shell script prompts you to enter the *`password_defined_during_DBSystem_creation`* and sets it in all shell scripts and SQL scripts that will be used in the practices.

```
$ chmod 777 /home/oracle/labs/update_pass.sh
$ /home/oracle/labs/update_pass.sh
dos2unix: converting file /home/oracle/labs/update_pass.sh to Unix
format ...
dos2unix: converting file /home/oracle/labs/M104785GC10/create_CDB20.sh
to Unix format ...
dos2unix: converting file /home/oracle/labs/M104781GC10/setup_DV.sh to
Unix format ...
...
Enter the password you set during the DBSystem creation:
password_defined_during_DBSystem_creation
$
```

Security Solutions

- Security
- Oracle Advanced Security
- Oracle Database Vault

Security

Force Upgraded Password File to be Case Sensitive

Starting in Oracle Database 20c, the parameter to enable or disable password file case sensitivity is removed. All passwords in new password files are case-sensitive.

Case-sensitive password files provide more security than older password files that are case insensitive. Oracle recommends that you use case-sensitive password files. However, upgraded password files from earlier Oracle Database releases can retain their original case-insensitivity. You can force your password files to be case-sensitive by migrating password files from one format to another.

- Practice: Forcing Upgraded Password File to be Case Sensitive
  This practice shows how the passwords in the password files in Oracle Database 20c are case-sensitive. In earlier Oracle Database releases, password files by default retain their original case-insensitive verifiers. The parameter to enable or disable password file case sensitivity `IGNORECASE` is removed. All passwords in new password files are case-sensitive.

Related Topics

- *Oracle® Database Security Guide*

Practice: Forcing Upgraded Password File to be Case Sensitive

This practice shows how the passwords in the password files in Oracle Database 20c are case-sensitive. In earlier Oracle Database releases, password files by default retain their original case-insensitive verifiers. The parameter to enable or disable password file case sensitivity `IGNORECASE` is removed. All passwords in new password files are case-sensitive.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Display the password file format of `CDB20`.

```
$ export ORACLE_BASE=/u01/app/oracle
$ cd $ORACLE_BASE/dbs
$ ls -l orapwCDB20
-rw-r----- 1 oracle oinstall 2048 Mar  5 09:48 orapwCDB20
$ orapwd describe file=orapwCDB20
Password file Description : format=12
$
```

3. Change SYS password and verify that the password is now case-sensitive.
   a. Change the SYS user password in the password file.

```
$ orapwd file=$ORACLE_BASE/dbs/orapwCDB20 sys=Y force=Y format=12
ignorecase=Y
Usage 1: orapwd file=<fname> force={y|n} asm={y|n}
          dbuniquename=<dbname> format={12|12.2}
          delete={y|n} input_file=<input-fname>
          'sys={y | password | external(<sys-external-name>)
                | global(<sys-directory-DN>)}'
          'sysbackup={y | password | external(<sysbackup-external-
name>)
                      | global(<sysbackup-directory-DN>)}'
          'sysdg={y | password | external(<sysdg-external-name>)
                  | global(<sysdg-directory-DN>)}'
          'syskm={y | password | external(<syskm-external-name>)
                  | global(<syskm-directory-DN>)}'

Usage 2: orapwd describe file=<fname>

  where
    file   - name of password file (required),
    password
           - password for SYS will be prompted
             if not specified at command line.
             Ignored, if input_file is specified,
    force  - whether to overwrite existing file, also clears
             CRS resource if it already has password file
             registered (optional),
    asm    - indicates that the ASM instance password file is to
             be stored in Automatic Storage Management (ASM)
             disk group (optional),
    dbuniquename
           - unique database name used to identify database
             password files residing in ASM diskgroup
             or Exascale Vault.
             Ignored when asm option is specified (optional),
    format - use format=12 for new 12c features like SYSBACKUP,
SYSDG
             and SYSKM support, longer identifiers, SHA2 Verifiers
etc.
             use format=12.2 for 12.2 features like enforcing user
             profile (password limits and password complexity) and
             account status for administrative users.
             If not specified, format=12.2 is default (optional),
    delete - drops a password file. Must specify 'asm',
             'dbuniquename' or 'file'. If 'file' is specified,
             the file must be located on an ASM diskgroup
             or Exascale Vault,
    input_file
           - name of input password file, from where old user
             entries will be migrated (optional),
    sys    - specifies if SYS user is password, externally or
             globally authenticated.
             For external SYS, also specifies external name.
```

```
                For global SYS, also specifies directory DN.
                SYS={y | password} specifies if SYS user password needs
                to be changed when used with input_file,
    sysbackup
           - creates SYSBACKUP entry (optional).
             Specifies if SYSBACKUP user is password, externally or
             globally authenticated.
             For external SYSBACKUP, also specifies external name.
             For global SYSBACKUP, also specifies directory DN.
             Ignored, if input_file is specified,
    sysdg  - creates SYSDG entry (optional).
             Specifies if SYSDG user is password, externally or
             globally authenticated.
             For external SYSDG, also specifies external name.
             For global SYSDG, also specifies directory DN.
             Ignored, if input_file is specified,
    syskm  - creates SYSKM entry (optional).
             Specifies if SYSKM user is password, externally or
             globally authenticated.
             For external SYSKM, also specifies external name.
             For global SYSKM, also specifies directory DN.
            Ignored, if input_file is specified,
    describe
           - describes the properties of specified password file
             (required).

  There must be no spaces around the equal-to (=) character.
$
```

The usage notes mention all possibles parameters that can be used in the command. IGNORECASE is not mentioned because it is now a deprecated parameter.

b. Re-enter the command without the deprecated parameter.

```
$ orapwd file=$ORACLE_BASE/dbs/orapwCDB20 sys=Y force=Y format=12
Enter password for SYS: password
$
```

c. Log on as SYS to CDB20.

```
$ sqlplus sys@CDB20 AS SYSDBA

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Dec 23 09:44:55
2019
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password_with_case-sensitiveness

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0
```

```
SQL> CONNECT sys@CDB20 AS SYSDBA

Enter password: password_without_case-sensitiveness
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to ORACLE.
SQL>
```

d. Display the list of the users.

```
SQL> CONNECT sys@CDB20 AS SYSDBA
Enter password: password_with_case-sensitiveness
Connected.
SQL> SET PAGES 100
SQL> COL username FORMAT A30
SQL> SELECT username, password_versions FROM dba_users ORDER BY 2,1;

USERNAME                       PASSWORD_VERSIONS
------------------------------ ----------------
SYS                            11G 12C
SYSTEM                         11G 12C
ANONYMOUS
APPQOSSYS
AUDSYS
CTXSYS
...
SQL> EXIT
$
```

Predefined Unified Audit Policies for Security Technical Implementation Guides (STIG) Compliance

Starting with this release, you can audit for Security Technical Implementation Guide (STIG) compliance by using new predefined unified audit policies.

These policies are as follows:

- `ORA_STIG_RECOMMENDATIONS`
- `ORA_ALL_TOPLEVEL_ACTIONS`
- `ORA_LOGON_LOGOFF`
- Practice: Using Predefined Unified Audit Policies for STIG Compliance
  This practice shows how to use predefined unified audit policies to implement Security Technical Implementation Guides (STIG) audit requirements.

Related Topics

- *Oracle® Database Security Guide*

Practice: Using Predefined Unified Audit Policies for STIG Compliance

This practice shows how to use predefined unified audit policies to implement Security Technical Implementation Guides (STIG) audit requirements.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Connect to PDB20 as SYSTEM and verify which predefined unified audit policies are implemented.

```
$ sqlplus system@PDB20

Enter password: password
Connected.
SQL> SELECT DISTINCT policy_name FROM audit_unified_policies ORDER BY 1;

POLICY_NAME
--------------------------------------------------------------------------------
--------
ORA_ACCOUNT_MGMT
ORA_ALL_TOPLEVEL_ACTIONS
ORA_CIS_RECOMMENDATIONS
ORA_DATABASE_PARAMETER
ORA_DV_AUDPOL
ORA_DV_AUDPOL2
ORA_LOGON_FAILURES
ORA_LOGON_LOGOFF
ORA_RAS_POLICY_MGMT
ORA_RAS_SESSION_MGMT
ORA_SECURECONFIG
ORA_STIG_RECOMMENDATIONS

12 rows selected.

SQL>
```

   Observe the three new predefined unified audit policies implemented.

3. Are these policies enabled to satisfy STIG compliance?

```
SQL> SELECT * FROM audit_unified_enabled_policies
      WHERE policy_name IN
('ORA_ALL_TOPLEVEL_ACTIONS','ORA_LOGON_LOGOFF','ORA_STIG_RECOMMENDATIONS
');

no rows selected

SQL>
```

   None of these are enabled.

4. Before enabling any of these policies, understand which actions they would audit.

a. Verify the actions audited by ORA_STIG_RECOMMENDATIONS.

```
SQL> COL audit_option FORMAT A26
SQL> COL AUDIT_OPTION_TYPE FORMAT A16
SQL> COL OBJECT_SCHEMA FORMAT A4
SQL> COL OBJECT_NAME FORMAT A22
SQL> COL OBJECT_TYPE FORMAT A7
SQL> SELECT audit_option, audit_option_type, object_schema,
object_name, object_type
      FROM  audit_unified_policies
      WHERE policy_name = 'ORA_STIG_RECOMMENDATIONS';

AUDIT_OPTION              AUDIT_OPTION_TYP OBJE
OBJECT_NAME            OBJECT_
------------------------- ---------------- ----
---------------------- -------
ALTER SESSION             SYSTEM PRIVILEGE NONE
NONE                   NONE
CREATE TABLE              STANDARD ACTION  NONE
NONE                   NONE
DROP TABLE                STANDARD ACTION  NONE
NONE                   NONE
ALTER TABLE               STANDARD ACTION  NONE
NONE                   NONE
CREATE SYNONYM            STANDARD ACTION  NONE
NONE                   NONE
DROP SYNONYM              STANDARD ACTION  NONE
NONE                   NONE
CREATE VIEW               STANDARD ACTION  NONE
NONE                   NONE
DROP VIEW                 STANDARD ACTION  NONE
NONE                   NONE
CREATE PROCEDURE          STANDARD ACTION  NONE
NONE                   NONE
ALTER PROCEDURE           STANDARD ACTION  NONE
NONE                   NONE
ALTER DATABASE            STANDARD ACTION  NONE
NONE                   NONE
ALTER USER                STANDARD ACTION  NONE
NONE                   NONE
ALTER SYSTEM              STANDARD ACTION  NONE
NONE                   NONE
CREATE USER               STANDARD ACTION  NONE
NONE                   NONE
CREATE ROLE               STANDARD ACTION  NONE
NONE                   NONE
DROP USER                 STANDARD ACTION  NONE
NONE                   NONE
DROP ROLE                 STANDARD ACTION  NONE
NONE                   NONE
SET ROLE                  STANDARD ACTION  NONE
NONE                   NONE
CREATE TRIGGER            STANDARD ACTION  NONE
NONE                   NONE
ALTER TRIGGER             STANDARD ACTION  NONE
```

```
NONE                    NONE
DROP TRIGGER                STANDARD ACTION  NONE
NONE                    NONE
CREATE PROFILE              STANDARD ACTION  NONE
NONE                    NONE
DROP PROFILE                STANDARD ACTION  NONE
NONE                    NONE
ALTER PROFILE               STANDARD ACTION  NONE
NONE                    NONE
DROP PROCEDURE              STANDARD ACTION  NONE
NONE                    NONE
CREATE MATERIALIZED VIEW    STANDARD ACTION  NONE
NONE                    NONE
ALTER MATERIALIZED VIEW     STANDARD ACTION  NONE
NONE                    NONE
DROP MATERIALIZED VIEW      STANDARD ACTION  NONE
NONE                    NONE
CREATE TYPE                 STANDARD ACTION  NONE
NONE                    NONE
DROP TYPE                   STANDARD ACTION  NONE
NONE                    NONE
ALTER ROLE                  STANDARD ACTION  NONE
NONE                    NONE
ALTER TYPE                  STANDARD ACTION  NONE
NONE                    NONE
CREATE TYPE BODY            STANDARD ACTION  NONE
NONE                    NONE
ALTER TYPE BODY             STANDARD ACTION  NONE
NONE                    NONE
DROP TYPE BODY              STANDARD ACTION  NONE
NONE                    NONE
DROP LIBRARY                STANDARD ACTION  NONE
NONE                    NONE
ALTER VIEW                  STANDARD ACTION  NONE
NONE                    NONE
CREATE FUNCTION             STANDARD ACTION  NONE
NONE                    NONE
ALTER FUNCTION              STANDARD ACTION  NONE
NONE                    NONE
DROP FUNCTION               STANDARD ACTION  NONE
NONE                    NONE
CREATE PACKAGE              STANDARD ACTION  NONE
NONE                    NONE
ALTER PACKAGE               STANDARD ACTION  NONE
NONE                    NONE
DROP PACKAGE                STANDARD ACTION  NONE
NONE                    NONE
CREATE PACKAGE BODY         STANDARD ACTION  NONE
NONE                    NONE
ALTER PACKAGE BODY          STANDARD ACTION  NONE
NONE                    NONE
DROP PACKAGE BODY           STANDARD ACTION  NONE
NONE                    NONE
CREATE LIBRARY              STANDARD ACTION  NONE
NONE                    NONE
```

```
CREATE JAVA                    STANDARD ACTION  NONE
NONE                      NONE
ALTER JAVA                     STANDARD ACTION  NONE
NONE                      NONE
DROP JAVA                      STANDARD ACTION  NONE
NONE                      NONE
CREATE OPERATOR                STANDARD ACTION  NONE
NONE                      NONE
DROP OPERATOR                  STANDARD ACTION  NONE
NONE                      NONE
ALTER OPERATOR                 STANDARD ACTION  NONE
NONE                      NONE
CREATE SPFILE                  STANDARD ACTION  NONE
NONE                      NONE
ALTER SYNONYM                  STANDARD ACTION  NONE
NONE                      NONE
ALTER LIBRARY                  STANDARD ACTION  NONE
NONE                      NONE
DROP ASSEMBLY                  STANDARD ACTION  NONE
NONE                      NONE
CREATE ASSEMBLY                STANDARD ACTION  NONE
NONE                      NONE
ALTER ASSEMBLY                 STANDARD ACTION  NONE
NONE                      NONE
ALTER PLUGGABLE DATABASE       STANDARD ACTION  NONE
NONE                      NONE
CREATE LOCKDOWN PROFILE        STANDARD ACTION  NONE
NONE                      NONE
DROP LOCKDOWN PROFILE          STANDARD ACTION  NONE
NONE                      NONE
ALTER LOCKDOWN PROFILE         STANDARD ACTION  NONE
NONE                      NONE
ADMINISTER KEY MANAGEMENT      STANDARD ACTION  NONE
NONE                      NONE
ALTER DATABASE DICTIONARY      STANDARD ACTION  NONE
NONE                      NONE
GRANT                          STANDARD ACTION  NONE
NONE                      NONE
REVOKE                         STANDARD ACTION  NONE
NONE                      NONE
ALL                            OLS ACTION       NONE
NONE                      NONE
EXECUTE                        OBJECT ACTION    SYS
DBMS_SCHEDULER            PACKAGE
EXECUTE                        OBJECT ACTION    SYS
DBMS_JOB                  PACKAGE
EXECUTE                        OBJECT ACTION    SYS
DBMS_RLS                  PACKAGE
EXECUTE                        OBJECT ACTION    SYS
DBMS_REDACT               PACKAGE
EXECUTE                        OBJECT ACTION    SYS
DBMS_TSDP_MANAGE          PACKAGE
EXECUTE                        OBJECT ACTION    SYS
DBMS_TSDP_PROTECT         PACKAGE
EXECUTE                        OBJECT ACTION    SYS
```

```
DBMS_NETWORK_ACL_ADMIN PACKAGE

75 rows selected.

SQL>
```

The policy once enabled audits all major actions that could damage the security and the smooth running of the database, and also all Oracle Label Security actions. This result shows that you should enable the policy for all users.

b. Verify the actions audited by `ORA_ALL_TOPLEVEL_ACTIONS`.

```
SQL> COL audit_option FORMAT A6
SQL> COL object_name FORMAT A11
SQL> COL audit_only_toplevel FORMAT A22
SQL> SELECT audit_option, audit_option_type, object_schema,
object_name,
               object_type, audit_only_toplevel
      FROM  audit_unified_policies
      WHERE policy_name = 'ORA_ALL_TOPLEVEL_ACTIONS';

AUDIT_ AUDIT_OPTION_TYP OBJE OBJECT_NAME OBJECT_ AUDIT_ONLY_TOPLEVEL
------ ---------------- ---- ----------- -------
----------------------
ALL    STANDARD ACTION  NONE NONE        NONE    YES

SQL>
```

The policy once enabled audits all top level actions of privileged users on any object that could damage the security of the database. This result shows that you should enable the policy for all users.

c. Verify the actions audited by `ORA_LOGON_LOGOFF`.

```
SQL> COL audit_option FORMAT A6
SQL> COL object_name FORMAT A11
SQL> COL audit_only_toplevel FORMAT A22
SQL> SELECT audit_option, audit_option_type, object_schema,
object_name,
               object_type, audit_only_toplevel
      FROM  audit_unified_policies
      WHERE policy_name = 'ORA_LOGON_LOGOFF';

AUDIT_ AUDIT_OPTION_TYP OBJE OBJECT_NAME OBJECT_ AUDIT_ONLY_TOPLEVEL
------ ---------------- ---- ----------- -------
----------------------
LOGON  STANDARD ACTION  NONE NONE        NONE    NO
LOGOFF STANDARD ACTION  NONE NONE        NONE    NO

SQL>
```

The policy once enabled audits all connection and disconnections that could display unsecure connections to the database. This policy is required for both

the Center for Internet Security (CIS) and Security for Technical Implementation Guides (STIG) requirements.

d. Enable all three audit policies for all users.

```
SQL> AUDIT POLICY ORA_STIG_RECOMMENDATIONS;

Audit succeeded.

SQL> AUDIT POLICY ORA_ALL_TOPLEVEL_ACTIONS;

Audit succeeded.

SQL> AUDIT POLICY ORA_LOGON_LOGOFF;

Audit succeeded.

SQL> EXIT
$
```

SYSLOG Destination for Common Unified Audit Policies

Certain predefined columns of unified audit records from common unified audit policies can be written to the UNIX SYSLOG destination.

To enable this feature, you set UNIFIED_AUDIT_COMMON_SYSTEMLOG, a new CDB level init.ora parameter. This enhancement enables all audit records from common unified audit policies to be consolidated into a single destination.

This feature is available only on UNIX platforms, not Windows.

- Practice: SYSLOG Destination for Common Unified Audit Policies
  This practice shows how to enable all audit records from common unified audit policies to be consolidated into a single destination. The new initialization parameter used for the configuration is supported only on UNIX platforms and NOT available on Windows.

Related Topics

- *Oracle® Database Security Guide*

Practice: SYSLOG Destination for Common Unified Audit Policies

This practice shows how to enable all audit records from common unified audit policies to be consolidated into a single destination. The new initialization parameter used for the configuration is supported only on UNIX platforms and NOT available on Windows.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before configuring the SYSLOG destination for common unified audit policies to be consolidated into a single destination, execute the /home/oracle/labs/M104781GC10/setup_SYSLOG_audit.sh shell script against CDB20. The shell script

creates a common user C##TEST and commonly grants the common user the CREATE SESSION and CREATE TABLE privileges.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_SYSLOG_audit.sh

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:38:30 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production

Version 20.2.0.0.0

LSNRCTL for Linux: Version 20.0.0.0.0 - Production on 20-MAR-2020
04:38:57

Copyright (c) 1991, 2019, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.150.76.66)
(PORT=1521)))
The command completed successfully
/usr/bin/ar cr /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/
libknlopt.a /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/20.2.0/dbhome_1/bin

 - Linking Oracle
rm -f /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/oracle
...
LSNRCTL for Linux: Version 20.0.0.0.0 - Production on 20-MAR-2020
04:39:09

Copyright (c) 1991, 2019, Oracle.  All rights reserved.

Starting /u01/app/oracle/product/20.2.0/dbhome_1/bin/tnslsnr: please
wait...

TNSLSNR for Linux: Version 20.0.0.0.0 - Production
System parameter file is /u01/app/oracle/homes/OraDB20Home1/network/
admin/listener.ora
Log messages written to /u01/app/oracle/diag/tnslsnr/edcdr8p1/listener/
alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.150.76.66)
(PORT=1521)))
```

```
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.150.76.66)
(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 20.0.0.0.0 -
Production
Start Date                20-MAR-2020 04:39:09
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/homes/OraDB20Home1/network/
admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/edcdr8p1/
listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.150.76.66)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:39:09 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> STARTUP
...
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Dec 24 02:34:44 2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Tue Dec 24 2019 02:31:07 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> CREATE USER c##test IDENTIFIED BY password CONTAINER=ALL;

User created.

SQL> GRANT CREATE SESSION, CREATE TABLE TO c##test CONTAINER=ALL;

Grant succeeded.

SQL> EXIT
$
```

3. Create a common and a local audit policy at the CDB root in CDB20, and a local audit policy at the PDB level in PDB20.

   a. First create the common and the local audit policy at the CDB root in CDB20.

```
$ sqlplus / AS SYSDBA

Connected.
SQL> CREATE AUDIT POLICY pol_common ACTIONS create table
CONTAINER=ALL;

Audit policy created.

SQL> AUDIT POLICY pol_common;

Audit succeeded.

SQL> CREATE AUDIT POLICY pol_root ACTIONS insert;

Audit policy created.

SQL> AUDIT POLICY pol_root;

Audit succeeded.

SQL> COL policy_name FORMAT A18
SQL> COL audit_option FORMAT A18
SQL> SELECT policy_name, audit_option, common
      FROM AUDIT_UNIFIED_POLICIES
      WHERE policy_name like 'POL%';

POLICY_NAME        AUDIT_OPTION       COM
------------------ ------------------ ---
POL_COMMON         CREATE TABLE       YES
POL_ROOT           INSERT             NO

SQL>
```

   b. Create the local audit policy at the PDB level in PDB20.

```
SQL> CONNECT system@PDB20
Enter password: password
Connected.
SQL> CREATE AUDIT POLICY pol_pdb20 ACTIONS select;

Audit policy created.

SQL> AUDIT POLICY pol_pdb20;

Audit succeeded.

SQL>
```

c. Display the policy names, their actions and commonality.

```
SQL> COL policy_name FORMAT A18
SQL> COL audit_option FORMAT A18
SQL> SELECT policy_name, audit_option, common
      FROM AUDIT_UNIFIED_POLICIES
      WHERE policy_name like 'POL%';

POLICY_NAME        AUDIT_OPTION       COM
------------------ ------------------ ---
POL_COMMON         CREATE TABLE       YES
POL_PDB20          SELECT             NO

SQL>
```

4. Configure the SYSLOG destination for common unified audit policies to be consolidated into a single destination. The facility_clause refers to the facility to which you will write the audit trail records. Valid choices are USER and LOCAL. If you enter LOCAL, then optionally append 0–7 to designate a local custom facility for the SYSLOG records. priority_clause refers to the type of warning in which to categorize the record. Valid choices are NOTICE, INFO, DEBUG, WARNING, ERR,CRIT , ALERT, and EMERG.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET UNIFIED_AUDIT_COMMON_SYSTEMLOG='local0.info'
SCOPE=SPFILE;

System altered.

SQL>
```

5. Configure the SYSLOG destination for local unified audit policies to be consolidated into a single destination.

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET UNIFIED_AUDIT_COMMON_SYSTEMLOG='local1.warning'
SCOPE=SPFILE;
ALTER SYSTEM SET UNIFIED_AUDIT_COMMON_SYSTEMLOG='local1.warning'
SCOPE=SPFILE
*
ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable database

SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET UNIFIED_AUDIT_SYSTEMLOG='local1.warning'
SCOPE=SPFILE;

System altered.

SQL>
```

Observe that the UNIFIED_AUDIT_COMMON_SYSTEMLOG is a CDB level init.ora parameter.

6. Restart the database instance because the initialization parameter UNIFIED_AUDIT_COMMON_SYSTEMLOG has been set at the SPFILE scope.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area 1426061008 bytes
Fixed Size                  9565904 bytes
Variable Size             889192448 bytes
Database Buffers          520093696 bytes
Redo Buffers                7208960 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN;

Pluggable database altered.

SQL>
```

7. Before audited actions are recorded by the SYSLOG system, define the OS directories for the SYSLOG files to store the audited records. Open another terminal session.

   a. Log in a root.

```
$ sudo su
#
```

   b. Edit the /etc/rsyslog.conf configuration file and under the RULES section, add as many lines as different values defined in the CDB for SYSTEMLOG to specify related OS directories.

```
# vi /etc/rsyslog.conf
...
#### RULES ####
...
# Save boot messages also to boot.log
local7.*                                                /var/log/
boot.log

# Unified Audit Rules
local0.info             /var/log/root_common_audit_records.log
local1.warning          /var/log/root_audit_records.log
...
#
```

c. Restart the SYSLOG daemon.

```
# cd /etc/init.d
# service rsyslog restart
Redirecting to /bin/systemctl restart  rsyslog.service
#
```

8. In the `oracle` UNIX session, log on as the common user `C##TEST` to the CDB root and perform a CREATE TABLE operation followed by INSERT operation on the table created.

```
SQL> CONNECT c##test

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 05:44:04 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SESSION SET default_sharing = 'EXTENDED DATA';

Session altered.

SQL> CREATE TABLE test (id NUMBER, label VARCHAR2(10));

Table created.

SQL> INSERT INTO test VALUES (1,'A');

1 row created.COMMIT;

SQL> INSERT INTO test VALUES (2,'B');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

9. Back in the `root` UNIX session, check that a syslog entry is created in `/var/log/root_common_audit_records.log` file because an audit record for CREATE TABLE got generated due to the common audit policy `POL_COMMON`.

```
# cat /var/log/root_common_audit_records.log
Mar 20 08:51:55 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"1" STMTID:"8" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1"
```

```
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
#
```

The single entry corresponds to the CREATE TABLE action audited commonly because the POL_COMMON audit policy audits all CREATE TABLE statements in all containers. The INSERT action is not recorded in this log file because the audit policy that audits INSERT statements, POL_ROOT is enabled only locally in the CDB root.

10. Check that syslog entries are created in /var/log/root_audit_records.log file because audit records for INSERT got generated due to the local root audit policy POL_ROOT.

```
# cat /var/log/root_audit_records.log
Mar 20 08:51:55 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"1" STMTID:"8" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
Mar 20 08:51:58 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"2" STMTID:"9" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
Mar 20 08:52:11 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '215' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"3" STMTID:"10" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
#
```

The first entry corresponds to the CREATE TABLE action audited commonly and thus also locally in the CDB root. The second and third entries correspond to the two INSERT actions recorded in this log file because the audit policy POL_ROOT that audits INSERT statements is enabled locally in the CDB root.

11. Back in the oracle UNIX session, log on as the common user C##TEST to the PDB PDB20 and perform a CREATE TABLE operation followed by INSERT operation on the table created.

```
SQL> CONNECT c##test@PDB20
Enter password: password
Connected.
SQL> CREATE TABLE testpdb20 (id NUMBER, label VARCHAR2(10));

Table created.

SQL> INSERT INTO testpdb20 VALUES (1,'A');

1 row created.

SQL> INSERT INTO testpdb20 VALUES (2,'B');
```

```
1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
$
```

12. Back in the root UNIX session, check whether a syslog entry is created in /var/log/root_common_audit_records.log file.

```
# cat /var/log/root_common_audit_records.log
Mar 20 08:51:55 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"1" STMTID:"8" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
Mar 20 09:02:48 your_server journal: Oracle Unified Audit[16023]:
LENGTH: '218' TYPE:"4" DBID:"79515510" SESID:"3581432176" CLIENTID:""
ENTRYID:"2" STMTID:"7" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TESTPDB20"
PDB_GUID:"A12EDF03A4B47886E053424C960AD028"
#
```

The second entry corresponds to the CREATE TABLE action audited commonly because the common audit policy POL_COMMON audits all CREATE TABLE statements in all containers and thus in PDB20 too. No INSERT action is recorded in this log file because the audit policy POL_ROOT that audits INSERT statements is created only locally in the CDB root and not commonly in all containers.

13. Check whether syslog entries are created in /var/log/root_audit_records.log file.

```
# cat /var/log/root_audit_records.log
...
Mar 20 08:51:55 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"1" STMTID:"8" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
Mar 20 08:51:58 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '214' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"2" STMTID:"9" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
Mar 20 08:52:11 your_server journal: Oracle Unified Audit[9653]:
LENGTH: '215' TYPE:"4" DBID:"2739122757" SESID:"112109882" CLIENTID:""
ENTRYID:"3" STMTID:"10" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2"
RETCODE:"0" SCHEMA:"C##TEST" OBJNAME:"TEST"
PDB_GUID:"9DF89CC354CB1655E0538EE0E40A712F"
...
# exit
```

```
exit
$ exit
```

Although a local audit policy `POL_PDB20` in `PDB20` audits INSERT actions, no audit record is written in the SYSLOG file because SYSLOG records only actions executed at the CDB level.

Unified Audit Policies Enforced on the Current User

Starting with this release, unified audit policies are enforced on the current user who executes the SQL statement.

In previous releases, unified audit policies were enforced on the user who owned the top-level user session (that is, the login user session) in which the SQL statement is executed.

Scenarios in which the current user is different from the login user include but are not limited to the following:

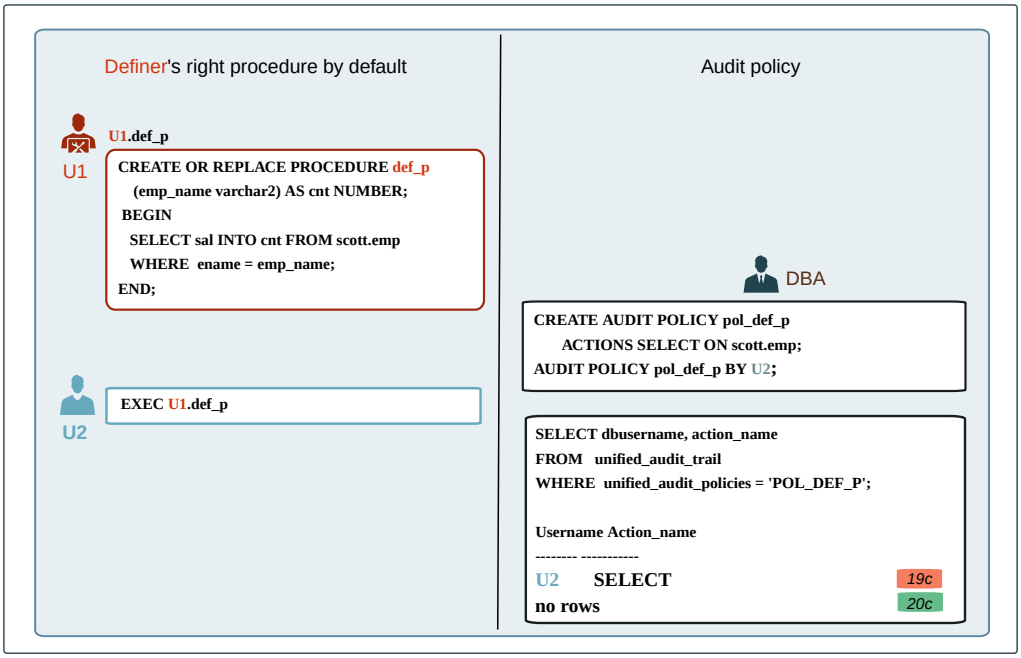
- Trigger execution
- Definer rights procedure execution
- Functions and procedures that are executed during the evaluation of views
- Details: Unified Audit Policies Enforced on the Current User
  This slide explains how unified audit policies are enforced on the user who owns the top-level user session that is, the login user session in which the SQL statement is executed.
- Practice: Enforcing Unified Audit Policies on the Current User
  This practice shows how unified audit policies are enforced on the current user who executes the SQL statement.

Related Topics

- *Oracle® Database Security Guide*

Details: Unified Audit Policies Enforced on the Current User

This slide explains how unified audit policies are enforced on the user who owns the top-level user session that is, the login user session in which the SQL statement is executed.



Starting with this release, unified audit policies are enforced on the current user who executes a SQL statement and not the login user.

Practice: Enforcing Unified Audit Policies on the Current User

This practice shows how unified audit policies are enforced on the current user who executes the SQL statement.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Use the `/home/oracle/labs/M104781GC10/setup_audit_policies.sh` shell script to create the `U1.PROCEMP` procedure in `PDB20`. The script also creates the users `U1` and `U2`.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_audit_policies.sh
...
SQL*Plus: Release 20.0.0.0.0 - Production on Fri Dec 13 05:55:55 2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> drop user u1 cascade;
drop user u1 cascade
          *
ERROR at line 1:
ORA-01918: user 'U1' does not exist

SQL> drop user u2 cascade;
drop user u2 cascade
```

```
          *
ERROR at line 1:
ORA-01918: user 'U2' does not exist

SQL> create user u1 identified by password;

User created.

SQL> grant create session, create procedure to u1;

Grant succeeded.

SQL> create user u2 identified by password;

User created.

SQL> grant select on hr.employees to u1, u2;

Grant succeeded.

SQL> grant create session to u2;

Grant succeeded.

SQL> grant select on unified_audit_trail to u1,u2;

Grant succeeded.

SQL>
SQL> CREATE OR REPLACE PROCEDURE u1.procemp (employee_id IN NUMBER)
  2  AS
  3     v_emp_id  NUMBER:=employee_id;
  4     v_sal NUMBER;
  5  BEGIN
  6     SELECT salary INTO v_sal FROM hr.employees WHERE
employee_id=v_emp_id;
  7     dbms_output.put_line('Salary is : '||v_sal || ' for Employee
ID: '||v_emp_id);
  8  END procemp;
  9  /

Procedure created.

SQL>
SQL> grant execute on u1.procemp to u2;

Grant succeeded.

SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. In PDB20, create and enable an audit policy so as to audit any query on HR.EMPLOYEES table executed by the login user U2.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Apr 3 14:44:59 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> CREATE AUDIT POLICY pol_emp ACTIONS select on hr.employees;

Audit policy created.

SQL> AUDIT POLICY pol_emp BY u2;

Audit succeeded.

SQL>
```

4. Connect to PDB20 as the user U2 and execute the U1.PROCEMP procedure.

```
SQL> CONNECT u2@PDB20
Enter password: password
SQL> SET SERVEROUTPUT ON
SQL> EXECUTE u1.procemp(206)
Salary is : 8300 for Employee ID: 206

PL/SQL procedure successfully completed.

SQL>
```

5. Display the DBUSERNAME (the login user) and the CURRENT_USER being the user who executed the procedure from the unified audit trail.

```
SQL> SELECT dbusername, current_user, action_name
     FROM   unified_audit_trail
     WHERE  unified_audit_policies = 'POL_EMP';

no rows selected

SQL> EXIT
$
```

> **Note:**
>
> Observe that the unified audit policy is enforced on the current user who executed the SQL statement, `U1`. Because only `U2` is audited and `U1` is the current user executing the query, there is no audit record generated that would give to the auditor the impression that the statement is executed by the user who owned the top-level user session.

Unified Audit Policy Configuration Changes Effective Immediately

Starting with this release, changes made to a unified audit policy become effective immediately in the current session and in all other on-going active sessions.

In previous releases, users who were affected by a changed unified audit policy had to log out of and then back into the session in order for the unified audit policy to take effect.

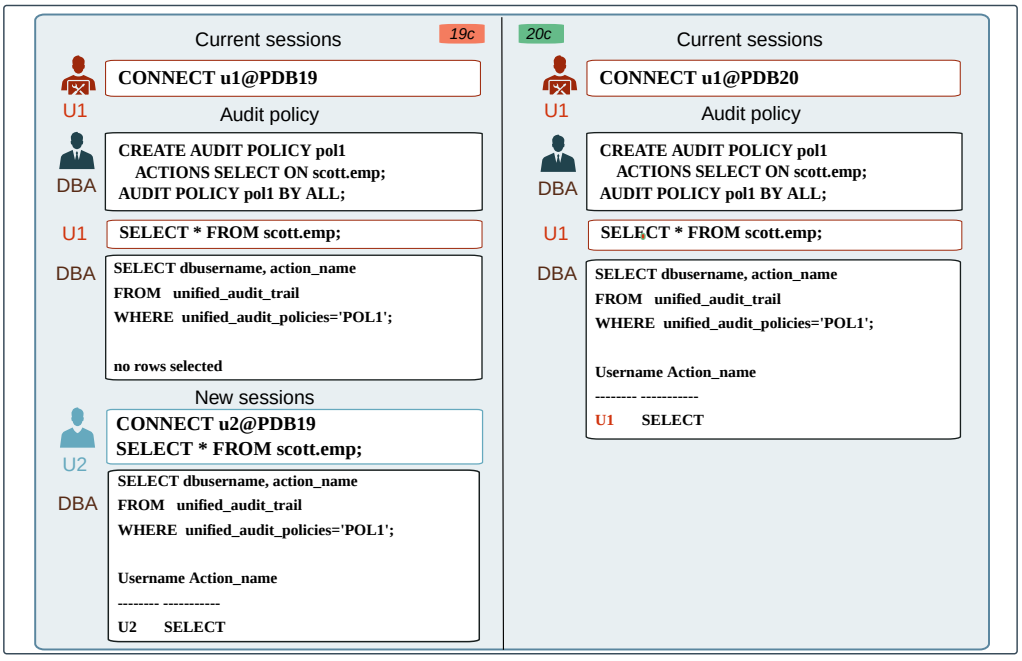
- Details: Unified Audit Policy Configuration Changes Effective Immediately
  This page explains how changes made to a unified audit policy become effective immediately in the current session and in all other on-going active sessions.
- Practice: Auditing Actions on Connected Sessions
  This practice shows how changes made to a unified audit policy become effective immediately in the current session and in all other on-going active sessions.

Related Topics

- *Oracle® Database Security Guide*

Details: Unified Audit Policy Configuration Changes Effective Immediately

This page explains how changes made to a unified audit policy become effective immediately in the current session and in all other on-going active sessions.



In previous releases, users who were affected by a changed unified audit policy had to log out of and then back into the session in order for the unified audit policy to take effect.

Practice: Auditing Actions on Connected Sessions

This practice shows how changes made to a unified audit policy become effective immediately in the current session and in all other on-going active sessions.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the `/home/oracle/labs/M104781GC10/setup_audit.sh` shell script.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_audit.sh

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:12:39 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

LSNRCTL for Linux: Version 20.0.0.0.0 - Production on 20-MAR-2020
04:13:03

Copyright (c) 1991, 2019, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.150.76.66)
(PORT=1521)))
The command completed successfully
/usr/bin/ar cr /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/
libknlopt.a /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/20.2.0/dbhome_1/bin

 - Linking Oracle
rm -f /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/oracle
...
LSNRCTL for Linux: Version 20.0.0.0.0 - Production on 20-MAR-2020
04:13:52

Copyright (c) 1991, 2019, Oracle.  All rights reserved.
```

```
Starting /u01/app/oracle/product/20.2.0/dbhome_1/bin/tnslsnr: please
wait...

TNSLSNR for Linux: Version 20.0.0.0.0 - Production
System parameter file is /u01/app/oracle/homes/OraDB20Home1/network/
admin/listener.ora
Log messages written to /u01/app/oracle/diag/tnslsnr/edcdr8p1/listener/
alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.150.76.66)
(PORT=1521)))
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.150.76.66)
(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 20.0.0.0.0 -
Production
Start Date                20-MAR-2020 04:13:52
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/homes/OraDB20Home1/network/
admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/edcdr8p1/
listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.150.76.66)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
The listener supports no services
The command completed successfully

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:13:52 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> STARTUP
...
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 05:09:56 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

specify password for HR as parameter 1:
```

```
specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:
...
SQL> BEGIN
  2  DBMS_AUDIT_MGMT.clean_audit_trail(
  3  audit_trail_type => DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL,
  4  use_last_arch_timestamp => false);
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL>
SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 05:09:55 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Mon Mar 09 2020 04:57:43 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL>
SQL> DROP USER u1 CASCADE;

User dropped.

SQL> DROP USER u2 CASCADE;

User dropped.

SQL> CREATE USER u1 identified by password;

User created.

SQL> GRANT create session TO u1;

Grant succeeded.

SQL> GRANT select ON hr.locations TO u1;

Grant succeeded.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
```

```
20.0.0.0.0 - Poduction
Version 20.2.0.0.0
$
```

3. Connect as U1 in to PDB20.

```
$ sqlplus u1@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:31:44 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL>
```

4. In another terminal session, connect as SYSTEM to PDB20 and create and enable an audit policy to audit any select on the HR.LOCATIONS table.
   a. Verify that Unified Auditing is enabled.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 20 04:32:22
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Fri Mar 20 2020 04:21:35 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> SELECT value FROM v$option WHERE parameter='Unified Auditing';

VALUE
----------------------------------------------------------------
TRUE

SQL>
```

   b. Create and enable an audit policy to audit any select on the HR.LOCATIONS table.

```
SQL> CREATE AUDIT POLICY pol1 ACTIONS SELECT ON hr.locations;
```

```
Audit policy created.

SQL> AUDIT POLICY pol1;

Audit succeeded.

SQL> SELECT dbusername, action_name FROM unified_audit_trail
WHERE  unified_audit_policies='POL1';

no rows selected.

SQL>
```

5. Back in the `U1` session, select rows from `HR.LOCATIONS` table.

```
SQL> SELECT street_address FROM hr.locations;

STREET_ADDRESS
----------------------------------------
1297 Via Cola di Rie
93091 Calle della Testa
2017 Shinjuku-ku
...

23  rows selected.

SQL> EXIT
$
```

6. Is the query executed by `U1` audited although not reconnected? Switch back in the `SYSTEM` session.

```
SQL> SELECT dbusername, action_name FROM unified_audit_trail
WHERE  unified_audit_policies='POL1';

DBUSERNAME
------------------------------------------------------------------------
--------
ACTION_NAME
----------------------------------------------------------------
U1
SELECT

SQL>
```

Observe the difference of behavior between the Oracle Database 19c and Oracle Database 20c: in Oracle Database 20c, enabled audit policies do not require already connected sessions to reconnect to get their actions be audited.

7. Drop the audit policy.

```
SQL> NOAUDIT POLICY pol1;

Noaudit succeeded.
```

```
SQL> DROP AUDIT POLICY pol1;

Audit Policy dropped.

SQL> EXIT
$
```

Oracle Blockchain Table

Blockchain tables are append-only tables in which only insert operations are allowed. Deleting rows is either prohibited or restricted based on time. Rows in a blockchain table are made tamper-resistant by special sequencing & chaining algorithms. Users can verify that rows have not been tampered. A hash value that is part of the row metadata is used to chain and validate rows.

Blockchain tables enable you to implement a centralized ledger model where all participants in the blockchain network have access to the same tamper-resistant ledger.

A centralized ledger model reduces administrative overheads of setting a up a decentralized ledger network, leads to a relatively lower latency compared to decentralized ledgers, enhances developer productivity, reduces the time to market, and leads to significant savings for the organization. Database users can continue to use the same tools and practices that they would use for other database application development.

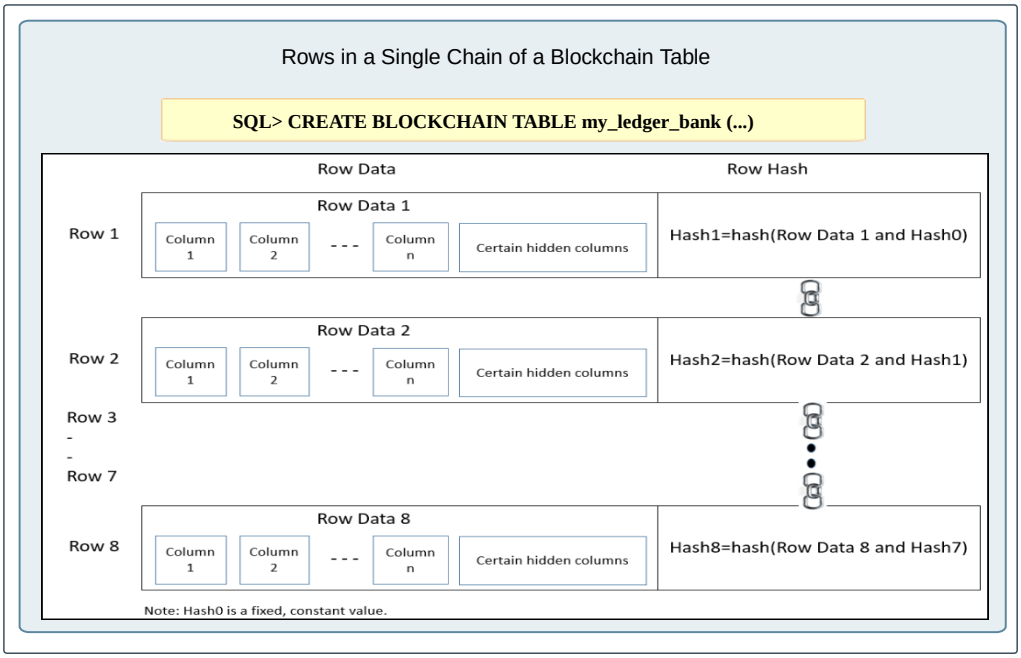
- Details: Oracle Blockchain Table
  Those pages provide more detailed information about blockchain tables and chained rows by row hash, how the blockchain tables are implemented, managed and how row data is handled in blockchain tables.
- Practice: Managing Blockchain Tables and Rows
  This practice shows how to create, alter and drop Oracle blockchain tables.

Related Topics

- *Oracle® Database Administrator's Guide*

Details: Oracle Blockchain Table

Those pages provide more detailed information about blockchain tables and chained rows by row hash, how the blockchain tables are implemented, managed and how row data is handled in blockchain tables.



Blockchain tables are used to implement centralized blockchain applications where the central authority is the Oracle Database. Centralized blockchains provide organizations with more customizability and control as they can decide who can participate in the network. The participants are different database users who trust Oracle Database to maintain a tamper-proof blockchain of transactions. All participants must have privileges to insert data into the blockchain table. The contents of the blockchain are defined and managed by the application. Compared to decentralized blockchains, centralized blockchains are useful in scenarios where a higher throughput and lower latency of transactions is preferred over consensus-based distributed blockchains.

Blockchain tables are insert-only tables that organize rows into a number of chains. Each row, except the first row in the chain, is chained to the previous row.

Rows in a blockchain table are tamper-proof. Each row contains a cryptographic hash value which is based on the data in that row and the hash value of the previous row in the chain. If a row is tampered with, the hash value of the row changes and this causes the hash value of the next row in the chain to change. An optional user signature can be added to a row for enhanced fraud detection.

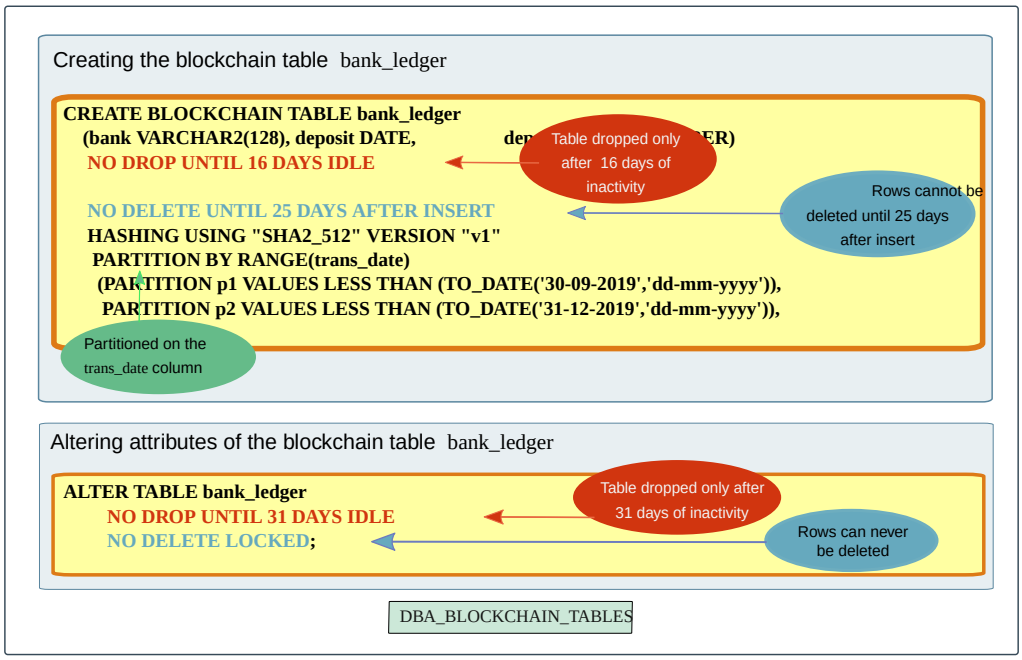
Use blockchain tables when immutability of data is critical for your centralized applications and you need to maintain a tamper-resistant ledger of current and historical transactions. A blockchain table is a building block. You must define the triggers or stored procedures required to perform the tasks that will implement a centralized blockchain. Information Lifecycle Management (ILM) is used to manage the lifecycle of data in blockchain tables. When the data in one or more partitions of a blockchain table is old, it can be moved to cheaper storage using ILM techniques.

Consider the following benefits of using blockchain tables:

- They provide application-transparent protection from frauds by other participants in the blockchain network.

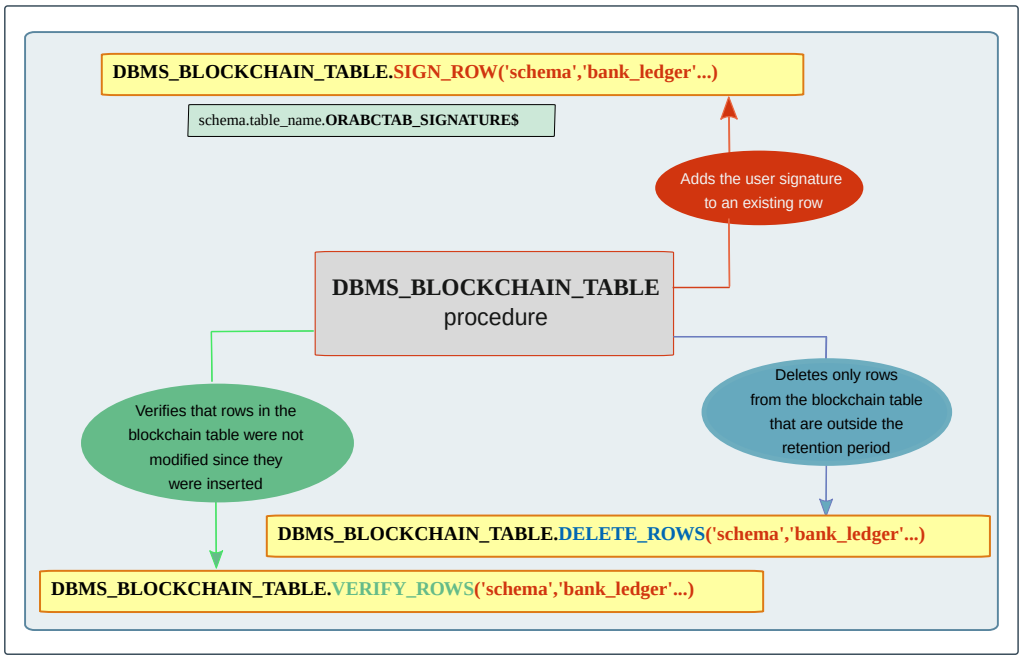
  Frauds can be detected by verifying rows in the blockchain table. This recomputes the hash value and verifies that it matches the value stored in the corresponding internal column.

- They do not need new infrastructure because they are part of Oracle database.
- They enable you to retain the current architecture and programming model. Therefore, existing database applications that have central authorities can be made more secure.
- They are easier to use compared to distributed blockchains.



Blockchain tables are append-only tables in which only insert operations are allowed. Deleting rows is either prohibited or restricted based on time. Rows in a blockchain table are made tamper-resistant by special sequencing & chaining algorithms. Users can verify that rows have not been tampered. A hash value that is part of the row metadata is used to chain and validate rows. Blockchain tables enable you to implement a centralized ledger model where all participants in the blockchain network have access to the same tamper-resistant ledger.

Blockchain tables can be indexed and partitioned. You can control whether and when rows are deleted from a blockchain table. You can also control whether the blockchain table can be dropped. Blockchain tables can be used along with (regular) tables in transactions and queries.



Signing Blockchain Table Rows

Signing a row sets a user signature for a previously created row. A signature provides additional security against tampering.

Oracle Database verifies that the current user owns the row being updated and the hash, if provided, matches the stored hash value of the row. You must have the `INSERT` privilege on the blockchain table. The existing signature of the row for which a signature is being added must be NULL. Use the `DBMS_BLOCKCHAIN_TABLE.SIGN_ROW` procedure to add a signature to an existing row.

Validating Data in Blockchain Tables

The PL/SQL procedure `DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS` verifies that rows in a blockchain table were not modified since they were inserted. Being tamper-proof is a key requirement for blockchain tables. You must have the `SELECT` privilege on the blockchain table to run this procedure.

You can validate all rows in the blockchain table or specify a criteria to filter rows that must be validated. Rows can be filtered using the instance ID, chain ID, or row creation time.

Deleting Rows in Blockchain Tables

Only rows that are outside the retention period can be deleted from a blockchain table. The PL/SQL procedure `DBMS_BLOCKCHAIN_TABLE.DELETE_ROWS` deletes all rows or rows that were created before a specified date.

Practice: Managing Blockchain Tables and Rows

This practice shows how to create, alter and drop Oracle blockchain tables.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Create the blockchain table named `AUDITOR.LEDGER_EMP` to maintain a tamper-resistant ledger of current and historical transactions about `HR.EMPLOYEES` in

PDB20. Rows can never be deleted in the blockchain table AUDITOR.LEDGER_EMP. Moreover the blockchain table can be dropped only after 31 days of inactivity.

a. Before starting creating the table, execute the /home/oracle/labs/M104781GC10/setup_user.sh shell script.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_user.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 05:34:10 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0


specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:


PL/SQL procedure successfully completed.
...
SQL> Disconnected from Oracle Database 20c Enterprise Edition
Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 05:34:16 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0
SQL> DROP USER auditor CASCADE;
DROP USER auditor CASCADE
          *
ERROR at line 1:
ORA-01918: user 'AUDITOR' does not exist


SQL> ALTER SYSTEM SET db_create_file_dest='/home/oracle/labs';

System altered.
```

```
SQL>
SQL> DROP TABLESPACE ledgertbs INCLUDING CONTENTS AND DATAFILES
cascade constraints;
DROP TABLESPACE ledgertbs INCLUDING CONTENTS AND DATAFILES cascade
constraints
*
ERROR at line 1:
ORA-00959: tablespace 'LEDGERTBS' does not exist

SQL> CREATE TABLESPACE ledgertbs;

Tablespace created.

SQL> CREATE USER auditor identified by password DEFAULT TABLESPACE
ledgertbs;

User created.

SQL> GRANT create session, create table, unlimited tablespace TO
auditor;

Grant succeeded.

SQL> GRANT execute ON sys.dbms_blockchain_table TO auditor;

Grant succeeded.

SQL> GRANT select ON hr.employees TO auditor;

Grant succeeded.

SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

b. Create the blockchain table named `AUDITOR.LEDGER_EMP`.

```
$ sqlplus auditor@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 05:37:25 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password

SQL> CREATE BLOCKCHAIN TABLE ledger_emp (employee_id NUMBER, salary
NUMBER);
CREATE BLOCKCHAIN TABLE ledger_emp (employee_id NUMBER, salary
NUMBER)
                                                            *
```

```
ERROR at line 1:
ORA-00905: missing keyword
```

> **Note:**
>
> Observe that the `CREATE BLOCKCHAIN TABLE` statement requires additional attributes.
> The `NO DROP`, `NO DELETE`, `HASHING USING`, and `VERSION` clauses are mandatory.

```
SQL> CREATE BLOCKCHAIN TABLE ledger_emp (employee_id NUMBER, salary
NUMBER)
                    NO DROP UNTIL 31 DAYS IDLE
                    NO DELETE LOCKED
                    HASHING USING "SHA2_512" VERSION "v1";

Table created.

SQL>
```

c. Verify the attributes set for the blockchain table in the appropriate data dictionary view.

```
SQL> SELECT row_retention, row_retention_locked,
                    table_inactivity_retention, hash_algorithm
              FROM   user_blockchain_tables
              WHERE  table_name='LEDGER_EMP';

ROW_RETENTION ROW TABLE_INACTIVITY_RETENTION HASH_ALG
------------- --- -------------------------- --------
              YES                         31 SHA2_512

SQL>
```

d. Show the description of the table.

```
SQL> DESC ledger_emp
 Name                                      Null?    Type
 ----------------------------------------- --------
----------------------------
 EMPLOYEE_ID                                        NUMBER
 SALARY                                             NUMBER

SQL>
```

> **Note:**
>
> Observe that the description displays only the visible columns.

e. Use the USER_TAB_COLS view to display all internal column names used to store internal information like the users number, the users signature.

```
SQL> COL "Data Length" FORMAT 9999
SQL> COL "Column Name" FORMAT A24
SQL> COL "Data Type" FORMAT A28
SQL> SELECT internal_column_id "Col ID", SUBSTR(column_name,1,30)
"Column Name",
                    SUBSTR(data_type,1,30) "Data Type",
data_length "Data Length"
              FROM   user_tab_cols
              WHERE  table_name = 'LEDGER_EMP' ORDER BY
internal_column_id;

    Col ID Column Name              Data Type
Data Length
---------- ------------------------ ---------------------------
-----------
         1 EMPLOYEE_ID
NUMBER                                22
         2 SALARY
NUMBER                                22
         3 ORABCTAB_INST_ID$
NUMBER                                22
         4 ORABCTAB_CHAIN_ID$
NUMBER                                22
         5 ORABCTAB_SEQ_NUM$
NUMBER                                22
         6 ORABCTAB_CREATION_TIME$  TIMESTAMP(6) WITH TIME
ZONE           13
         7 ORABCTAB_USER_NUMBER$
NUMBER                                22
         8 ORABCTAB_HASH$
RAW                                 2000
         9 ORABCTAB_SIGNATURE$
RAW                                 2000
        10 ORABCTAB_SIGNATURE_ALG$
NUMBER                                22
        11 ORABCTAB_SIGNATURE_CERT$
RAW                                   16
        12 ORABCTAB_SPARE$
RAW                                 2000

12 rows selected.

SQL>
```

3. Insert rows into the blockchain table as if your auditing application would do it.

   a. Insert a first row into the blockchain table.

```
SQL> INSERT INTO ledger_emp VALUES (106,12000);

1 row created.
```

```
SQL> COMMIT;

Commit complete.

SQL>
```

b. Display the internal values of the first row of the chain.

```
SQL> COL "Chain date" FORMAT A17
SQL> COL "Chain ID" FORMAT 99999999
SQL> COL "Seq Num" FORMAT 99999999
SQL> COL "User Num" FORMAT 9999999
SQL> COL "Chain HASH" FORMAT 99999999999999
SQL> SELECT ORABCTAB_CHAIN_ID$ "Chain ID", ORABCTAB_SEQ_NUM$ "Seq
Num",
            to_char(ORABCTAB_CREATION_TIME$,'dd-Mon-YYYY hh-mi')
"Chain date",
            ORABCTAB_USER_NUMBER$ "User Num", ORABCTAB_HASH$ "Chain
HASH"
     FROM   ledger_emp;

 Chain ID   Seq Num Chain date        User Num
--------- --------- ----------------- --------
Chain HASH
--------------------------------------------------------------------
------------
       14         1 06-Apr-2020 12-26      119
5812238B734B019EE553FF8A7FF573A14CFA1076AB312517047368D600984CFAB001
FA1FF2C98B13
9AB03DDCCF8F6C14ADF16FFD678756572F102D43420E69B3

SQL>
```

c. Connect as HR and insert a row into the blockchain table as if your auditing application would do it. First grant the INSERT privilege on the table to HR.

```
SQL> GRANT insert ON ledger_emp TO hr;

Grant succeeded.

SQL>
```

d. Connect as HR and insert a new row.

```
SQL> CONNECT hr@PDB20
Enter password: password
Connected.
SQL> INSERT INTO  auditor.ledger_emp VALUES (106,24000);

1 row created.

SQL> COMMIT;

Commit complete.
```

```
SQL>
```

e. Connect as AUDITOR and display the internal and external values of the blockchain table rows.

```
SQL> CONNECT auditor@PDB20
Enter password: password
Connected.
SQL> SELECT ORABCTAB_CHAIN_ID$ "Chain ID", ORABCTAB_SEQ_NUM$ "Seq
Num",
                to_char(ORABCTAB_CREATION_TIME$,'dd-Mon-YYYY hh-mi')
"Chain date",
                ORABCTAB_USER_NUMBER$ "User Num", ORABCTAB_HASH$
"Chain HASH",
                employee_id, salary
         FROM   ledger_emp;

 Chain ID   Seq Num Chain date        User Num
--------- --------- ---------------- --------
Chain HASH
------------------------------------------------------------------
------------
EMPLOYEE_ID     SALARY
----------- ----------
       14         1 06-Apr-2020 12-26      119
5812238B734B019EE553FF8A7FF573A14CFA1076AB312517047368D600984CFAB001
FA1FF2C98B13
9AB03DDCCF8F6C14ADF16FFD678756572F102D43420E69B3
        106      12000

       14         2 06-Apr-2020 12-28      118
BBCDACC41B489DFBD8E28244841411937BD716F987BE750146572C555311E377D6DB
A28D392C61E7
D75BA47BFCB3A2F4920A2C149409E89FBA63E10549DF4F47
        106      24000

SQL>
```

Observe that the user number is different. This value is the same value as V$SESSION.USER# column.

4. Delete the row inserted by HR.

```
SQL> DELETE FROM ledger_emp WHERE ORABCTAB_USER_NUMBER$ = 106;
DELETE FROM ledger_emp WHERE ORABCTAB_USER_NUMBER$ = 106
            *
ERROR at line 1:
ORA-05715: operation not allowed on the blockchain table

SQL>
```

You cannot delete rows in a blockchain table with the DML `DELETE` command. You must use the `DBMS_BLOCKCHAIN_TABLE` package.

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
   NUMBER_ROWS NUMBER;
BEGIN
   DBMS_BLOCKCHAIN_TABLE.DELETE_EXPIRED_ROWS('AUDITOR','LEDGER_EMP',
null, NUMBER_ROWS);
   DBMS_OUTPUT.PUT_LINE('Number of rows deleted=' || NUMBER_ROWS);
END;
/    2    3    4    5    6    7
Number of rows deleted=0

PL/SQL procedure successfully completed.
SQL>
```

You can delete rows in a blockchain table only by using the `DBMS_BLOCKCHAIN_TABLE` package, and only rows that are outside the retention period. This is the reason why the procedure successfully completes without deleting any row.

If the Oracle Database release installed is 20.0.0, then the procedure to use is `DBMS_BLOCKCHAIN_TABLE.DELETE_ROWS` and not `DBMS_BLOCKCHAIN_TABLE.DELETE_EXPIRED_ROWS`.

5. Truncate the table.

```
SQL> TRUNCATE TABLE ledger_emp;
TRUNCATE TABLE ledger_emp
               *
ERROR at line 1:
ORA-05715: operation not allowed on the blockchain table

SQL>
```

6. Specify now that rows cannot be deleted until 15 days after they were created.

```
SQL> ALTER TABLE ledger_emp NO DELETE UNTIL 15 DAYS AFTER INSERT;
ALTER TABLE ledger_emp NO DELETE UNTIL 15 DAYS AFTER INSERT
*
ERROR at line 1:
ORA-05731: blockchain table LEDGER_EMP cannot be altered

SQL>
```

Why cannot you change this attribute?

You created the table with the `NO DELETE LOCKED` attribute. The `LOCKED` clause indicates that you can never subsequently modify the row retention.

7. Drop the table.

```
SQL> DROP TABLE ledger_emp;
DROP TABLE ledger_emp
```

```
           *
ERROR at line 1:
ORA-05723: drop blockchain table LEDGER_EMP not allowed

SQL>
```

> **Note:**
>
> Observe that the error message is slightly different. The error message from the two previous commands explained that the operation was not possible on a blockchain table. The current error message explains that the `DROP TABLE` is not possible but on this `LEDGER_EMP` table.
> The blockchain table was created so that it cannot be dropped before 31 days of inactivity.

8. Change the behavior of the table to allow a lower retention.

```
SQL> ALTER TABLE ledger_emp NO DROP UNTIL 1 DAYS IDLE;
ALTER TABLE auditor.ledger_emp NO DROP UNTIL 1 DAYS IDLE
*
ERROR at line 1:
ORA-05732: retention value cannot be lowered

SQL> ALTER TABLE ledger_emp NO DROP UNTIL 40 DAYS IDLE;

Table altered.

SQL>
```

You can only increase the retention value. This prohibits the possibility to drop and remove any historical information that needs to be kept for security purposes.

9. Create another blockchain table `AUDITOR.LEDGER_TEST`. Rows cannot be deleted until 5 days after they were inserted, allowing rows to be deleted. Moreover the blockchain table can be dropped only after 1 day of inactivity, but to .

   a. Create the blockchain table.

```
SQL> CREATE BLOCKCHAIN TABLE auditor.ledger_test (id NUMBER, label
VARCHAR2(2))
       NO DROP UNTIL 1 DAYS IDLE
       NO DELETE UNTIL 5 DAYS AFTER INSERT
       HASHING USING "SHA2_512" VERSION "v1";
 2    3    4  CREATE BLOCKCHAIN TABLE auditor.ledger_test (id
NUMBER, label VARCHAR2(2))
*
ERROR at line 1:
ORA-05741: minimum retention time too low, should be at least 16
days

SQL> CREATE BLOCKCHAIN TABLE auditor.ledger_test (id NUMBER, label
VARCHAR2(2))
       NO DROP UNTIL 16 DAYS IDLE
```

```
      NO DELETE UNTIL 16 DAYS AFTER INSERT
      HASHING USING "SHA2_512" VERSION "v1";

Table created.

SQL>
```

b. Connect as HR and insert a row into the blockchain table as if your auditing application would do it. First grant the INSERT privilege on the table to HR.

```
SQL> GRANT insert ON auditor.ledger_test TO hr;

Grant succeeded.

SQL>
```

c. Connect as HR and insert a new row.

```
SQL> CONNECT hr@PDB20
Enter password: password
Connected.
SQL> INSERT INTO auditor.ledger_test VALUES (1,'A1');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

d. Connect as AUDITOR and display the row inserted.

```
SQL> CONNECT auditor@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM auditor.ledger_test;

        ID LA
---------- --
         1 A1

SQL>
```

10. Regularly verify that the content of the rows are still valid.

- Use the DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS to validate the rows.

```
SQL> CONNECT auditor@PDB20
Enter password: password
Connected.
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
   row_count NUMBER;
   verify_rows NUMBER;
```

```
   instance_id NUMBER;
BEGIN
  FOR instance_id IN 1 .. 2 LOOP
    SELECT COUNT(*) INTO row_count FROM auditor.ledger_test WHERE
ORABCTAB_INST_ID$=instance_id;
    DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS('AUDITOR','LEDGER_TEST',
NULL, NULL, instance_id, NULL, verify_rows);
    DBMS_OUTPUT.PUT_LINE('Number of rows verified in instance Id
'|| instance_id || ' = '|| row_count);
  END LOOP;
END;
/
Number of rows verified in instance Id 1 = 1
Number of rows verified in instance Id 2 = 0

PL/SQL procedure successfully completed.
SQL> EXIT
$
```

Oracle Advanced Security

- Ability to Set the Default Tablespace Encryption Algorithm

Ability to Set the Default Tablespace Encryption Algorithm

You now can set the `TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM` dynamic parameter to define the default encryption algorithm for tablespace creation operations.

For example, if you set `TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM` to `AES256`, then future tablespace creation operations will use `AES256` as the default encryption algorithm. `TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM` applies to both offline and online tablespace encryption operations. In addition, when you create a new tablespace using Database Configuration Assistant (DBCA), you can set the default tablespace encryption algorithm by using the DBCA command line for silent installations.

Supported encryption algorithms are `AES128`, `AES192`, `AES256`, and `3DES168` If you do not set `TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM`, then the default encryption algorithm is the default that was used in previous releases: `AES128`.

- Practice: Setting the Default Tablespace Encryption Algorithm
  This practice shows how to define the default tablespace encryption algorithm for tablespace creation operations by setting a dynamic parameter.

Related Topics

- *Oracle® Database Advanced Security Guide*

Practice: Setting the Default Tablespace Encryption Algorithm

This practice shows how to define the default tablespace encryption algorithm for tablespace creation operations by setting a dynamic parameter.

1. Before starting any new practice, refer to the Oracle Cloud. Be aware that encryption is configured by default in Oracle Database Cloud.

2. Connect to the CDB root and display the default tablespace encryption algorithm.

```
$ sqlplus / AS SYSDBA
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Apr 1 08:09:44 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL> SHOW PARAMETER TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM

NAME                                 TYPE   VALUE
------------------------------------ ------
----------------------
tablespace_encryption_default_algorithm    string AES128
SQL>
```

3. Change the tablespace encryption algorithm.

```
SQL> ALTER SYSTEM SET TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM=AES192;

System altered.

SQL>
```

4. Connect to the PDB and create a new tablespace in PDBTEST.

```
SQL> ALTER SESSION SET CONTAINER=PDB20;

Session altered.

SQL> CREATE TABLESPACE tbstest DATAFILE 'test01.dbf' SIZE 2M;

Tablespace created.

SQL>
```

5. Verify the tablespace encryption algorithm used for TBSTEST.

```
SQL> SELECT name, encryptionalg
              FROM v$tablespace t, v$encrypted_tablespaces v
              WHERE t.ts#=v.ts#;

NAME                           ENCRYPT
------------------------------ -------
USERS                          AES128
TBSTEST                        AES192

SQL> EXIT
$
```

Oracle Database Vault

Ability to Prevent Local Oracle Database Vault Policies from Blocking Common Operations

Starting with this release, a `DV_OWNER` common user in the CDB root can prevent local users from creating Oracle Database Vault controls on common objects in a pluggable database (PDB).

Blocking common users from common operations can prevent the execution of SQL commands that are necessary for managing the application or CDB database. To prevent this situation, a user who has the `DV_OWNER` role in the root can execute the `DBMS_MACADM.ALLOW_COMMON_OPERATION` procedure to control whether local PDB users can create Database Vault controls on common users' objects (database or application).

In previous releases, in a multitenant environment, a local Oracle Database Vault user could create Database Vault policies that could potentially block application or common operations. Blocking common users from common operations can prevent the execution of SQL commands that are necessary for managing the application or CDB database. To prevent this situation, a user who has the `DV_OWNER` role in the root can execute the `DBMS_MACADM.ALLOW_COMMON_OPERATION` procedure to control whether local PDB users can create Database Vault controls on common users' objects (database or application).

- Practice: Preventing Local Users from Blocking Common Operations - Realms
  This practice shows how to prevent local users from creating Oracle Database Vault controls on common users objects which would prevent common users from accessing local data in their own schema in PDBs. A PDB local Database Vault Owner can create a realm around common Oracle schemas like `DVSYS` or `CTXSYS` and prevent it functioning correctly. For the purposes of this practice, the `C##TEST1` custom schema is created in CDB root to show this feature.
- Practice: Preventing Local Users from Blocking Common Operations - Command Rules
  This practice shows how to prevent local users from creating Oracle Database Vault controls on common users which would prevent them from performing commands on their own objects or even from logging in to the PDB in which their objects reside.

Related Topics

- *Oracle® Database Vault Administrator's Guide*

Practice: Preventing Local Users from Blocking Common Operations - Realms

This practice shows how to prevent local users from creating Oracle Database Vault controls on common users objects which would prevent common users from accessing local data in their own schema in PDBs. A PDB local Database Vault Owner can create a realm around common Oracle schemas like `DVSYS` or `CTXSYS` and prevent it functioning correctly. For the purposes of this practice, the `C##TEST1` custom schema is created in CDB root to show this feature.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the `/home/oracle/labs/M104781GC10/setup_DV.sh` shell script. The shell script configures and enables Database Vault at the CDB root level, creates the `HR.G_EMP` table in the root container, configures and enables Database Vault at the PDB level, and creates the `HR.L_EMP` table in `PDB20`.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_DV.sh
$ ./setup_DV_CDB.sh

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Feb 19 05:38:54 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> drop user c##sec_admin cascade;
drop user c##sec_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'C##SEC_ADMIN' does not exist


SQL> create user c##sec_admin identified by password container=ALL;

User created.

SQL> grant create session, set container, restricted session, DV_OWNER
to c##sec_admin container=ALL;

Grant succeeded.

SQL> drop user c##accts_admin cascade;
drop user c##accts_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'C##ACCTS_ADMIN' does not exist


SQL> create user c##accts_admin identified by password container=ALL;

User created.

SQL> grant create session, set container, DV_ACCTMGR to c##accts_admin
container=ALL;

Grant succeeded.
```

```
SQL> grant select on sys.dba_dv_status to c##accts_admin container=ALL;

Grant succeeded.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
...
Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Tue Feb 18 2020 08:26:21 +00:00

SQL> DROP TABLE g_emp;

Table dropped.

SQL> CREATE TABLE g_emp(name CHAR(10), salary NUMBER) ;

Table created.

SQL> INSERT INTO g_emp values('EMP_GLOBAL',1000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 18 08:27:58 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Tue Feb 18 2020 08:27:54 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE l_emp;

Table dropped.

SQL> CREATE TABLE l_emp(name CHAR(10), salary NUMBER);

Table created.

SQL> INSERT INTO l_emp values('EMP_LOCAL',2000);
```

```
1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 18 08:27:58 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Tue Feb 18 2020 08:27:54 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE l_tab;

Table dropped.

SQL> CREATE TABLE l_tab(code NUMBER);

Table created.

SQL> INSERT INTO l_tab values(1);

1 row created.

SQL> INSERT INTO l_tab values(2);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Connect to the CDB root as `C##SEC_ADMIN` to verify the status of `DV_ALLOW_COMMON_OPERATION`. This is the default behavior: it allows local users to create Database Vault controls on common users objects.

```
$ sqlplus c##sec_admin
Enter password: password
```

```
SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                       STATU
-------------------------- -----
DV_ALLOW_COMMON_OPERATION FALSE

SQL>
```

4. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when there is no realm applied on C##TEST1 objects.

   a. Connect to the CDB root as C##TEST1, the table common owner.

   ```
   SQL> CONNECT c##test1
   Enter password: password
   Connected.
   SQL> SELECT * FROM c##test1.g_emp;

   NAME           SALARY
   ---------- ----------
   EMP_GLOBAL       1000

   SQL>
   ```

   b. Connect to the CDB root as C##TEST2, another common user.

   ```
   SQL> CONNECT c##test2
   Enter password: password
   Connected.
   SQL> SELECT * FROM c##test1.g_emp;

   NAME           SALARY
   ---------- ----------
   EMP_GLOBAL       1000

   SQL>
   ```

   c. Connect to PDB20 as C##TEST1, the table common owner.

   ```
   SQL> CONNECT c##test1@PDB20
   Enter password: password
   Connected.
   SQL> SELECT * FROM c##test1.l_emp;

   NAME           SALARY
   ---------- ----------
   EMP_LOCAL        2000

   SQL>
   ```

   d. Connect to PDB20 as C##TEST2, another common user.

   ```
   SQL> CONNECT c##test2@PDB20
   Enter password: password
   ```

```
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

5. Test how data is accessible in both the table `HR.G_EMP` in the CDB root and the table `C##TEST1.L_EMP` in `PDB20` when a regular realm is applied on `C##TEST1` objects in the CDB root.

   a. Create a common regular realm on `C##TEST1` tables in the CDB root.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Root Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 0);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Root Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

   b. Connect to the CDB root as `C##TEST1`, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Root Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

6. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a mandatory realm is applied on C##TEST1 objects in the CDB root.

a. Create a common mandatory realm on C##TEST1 tables in the CDB root.

```
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Root Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 1);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Root Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Root Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

7. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a PDB regular realm is applied on C##TEST1 objects in PDB20.

a. Create a PDB regular realm on C##TEST1 tables in PDB20.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 0);
END;
/  2    3    4    5    6    7    8    9
```

```
PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;
SELECT * FROM c##test1.l_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

f. Drop the realm.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

8. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a PDB mandatory realm is applied on C##TEST1 objects in PDB20.

a. Create a PDB mandatory realm on C##TEST1 tables in PDB20.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 1);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.
```

```
SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;
SELECT * FROM c##test1.l_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;
SELECT * FROM c##test1.l_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

f. Drop the realm.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

9. Connect to the CDB root as C##SEC_ADMIN and restrict local users from creating Oracle Database Vault controls on common user C##TEST1 objects. Set DV_ALLOW_COMMON_OPERATION to TRUE.

```
SQL CONNECT c##sec_admin
Enter password: password
Connected.
SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION FALSE

SQL> EXEC DBMS_MACADM.ALLOW_COMMON_OPERATION

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION TRUE

SQL>
```

10. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a regular realm is applied on C##TEST1 objects in the CDB root.

a. Create a common regular realm on C##TEST1 tables in the CDB root.

```
SQL CONNECT c##sec_admin
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Root Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 0);
END;
/  2    3    4    5    6    7    8    9
```

```
PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Root Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Root Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

11. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a mandatory realm is applied on C##TEST1 objects in the CDB root.

a. Create a common mandatory realm on C##TEST1 tables in the CDB root.

```
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Root Test Realm',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 1);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Root Test Realm',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;
SELECT * FROM c##test1.g_emp
                       *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Root Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

12. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a PDB regular realm is applied on C##TEST1 objects in PDB20.

a. Create a PDB regular realm on C##TEST1 tables in PDB20.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Test Realm1',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 0);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Test Realm1',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8
BEGIN
*
ERROR at line 1:
ORA-47286: cannot add %, C##TEST1.%  to a realm
ORA-06512: at "DVSYS.DBMS_MACADM", line 1059
ORA-06512: at line 2


SQL> !oerr ora 47286
47286, 00000, "cannot add %s, %s.%s  to a realm"
// *Cause: When ALLOW COMMON OPERATION was set to TRUE, a smaller
scope user was not allowed to add a larger scope user's object or a
larger scope role to a realm.
// *Action: When ALLOW COMMON OPERATION is TRUE, do not add a
larger scope user's object or a larger scope role to a realm.
```

```
SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm1')

PL/SQL procedure successfully completed.

SQL>
```

13. Test how data is accessible in both the table C##TEST1.G_EMP in the CDB root and the table C##TEST1.L_EMP in PDB20 when a PDB mandatory realm is applied on C##TEST1 objects in PDB20.

a. Create a PDB mandatory realm on C##TEST1 tables in PDB20.

```
SQL> BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Test Realm1',
  description   => 'Test Realm description',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
  realm_type    => 1);
END;
/  2    3    4    5    6    7    8    9

PL/SQL procedure successfully completed.

SQL> BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Test Realm1',
  object_owner => 'C##TEST1',
  object_name  => '%',
  object_type  => '%');
END;
/  2    3    4    5    6    7    8
BEGIN
*
ERROR at line 1:
ORA-47286: cannot add %, C##TEST1.%  to a realm
ORA-06512: at "DVSYS.DBMS_MACADM", line 1059
ORA-06512: at line 2

SQL>
```

b. Connect to the CDB root as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
```

```
EMP_GLOBAL       1000

SQL>
```

c. Connect to the CDB root as C##TEST2, another common user.

```
SQL> CONNECT c##test2
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.g_emp;

NAME           SALARY
---------- ----------
EMP_GLOBAL       1000

SQL>
```

d. Connect to PDB20 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

e. Connect to PDB20 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB20
Enter password: password
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME           SALARY
---------- ----------
EMP_LOCAL        2000

SQL>
```

f. Drop the realm.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm1')

PL/SQL procedure successfully completed.

SQL>
```

14. Disable Database Vault in both the PDB and the CDB root.

```
$ /home/oracle/labs/M104781GC10/disable_DV.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:25:59 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Mon Apr 06 2020 15:23:56 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:00 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Mon Apr 06 2020 15:23:58 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:02 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> shutdown immediate
Database closed.
```

```
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:23 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area 6442447392 bytes
Fixed Size                  9581088 bytes
Variable Size            1090519040 bytes
Database Buffers         5318377472 bytes
Redo Buffers               23969792 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

Let's summarize the behavior of data access on common users objects in PDBs when you switch the `DV_ALLOW_COMMON_OPERATION` value.

FALSE			**TRUE**	
	C##TEST1	C##TEST2	C##TEST1	C##TEST2
Common Regular or Mandatory Realm in CDB root	No change	No change	No change	No change
PDB Regular Realm	Access	Blocked	Access	**Access**
PDB Mandatory Realm	Blocked	Blocked	**Access**	**Access**

If you create a regular or mandatory realm in the CDB root and a regular or mandatory PDB realm, and if DV_ALLOW_COMMON_OPERATION is TRUE, then data of common users objects is accessible.

If local realms had been created when DV_ALLOW_COMMON_OPERATION was set to FALSE, they would still exist after the new control but enforcement would be ignored.

Practice: Preventing Local Users from Blocking Common Operations - Command Rules

This practice shows how to prevent local users from creating Oracle Database Vault controls on common users which would prevent them from performing commands on their own objects or even from logging in to the PDB in which their objects reside.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the /home/oracle/labs/M104781GC10/setup_DV_CR.sh shell script. The shell script configures and enables Database Vault at the CDB root level and at the PDB level, and creates the C##TEST1 and C##TEST2 common users.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_DV_CR.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Feb 19 05:38:54 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> drop user c##sec_admin cascade;
drop user c##sec_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'C##SEC_ADMIN' does not exist


SQL> create user c##sec_admin identified by password container=ALL;

User created.

SQL> grant create session, set container, restricted session, DV_OWNER
to c##sec_admin container=ALL;

Grant succeeded.

SQL> drop user c##accts_admin cascade;
drop user c##accts_admin cascade
          *
ERROR at line 1:
ORA-01918: user 'C##ACCTS_ADMIN' does not exist
```

```
SQL> create user c##accts_admin identified by password container=ALL;

User created.

SQL> grant create session, set container, DV_ACCTMGR to c##accts_admin
container=ALL;

Grant succeeded.

SQL> grant select on sys.dba_dv_status to c##accts_admin container=ALL;

Grant succeeded.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
...
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Feb 19 11:14:29 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> GRANT dba to c##test1 CONTAINER=ALL;

Grant succeeded.

...
Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE l_tab;

Table dropped.

SQL> CREATE TABLE l_tab(code NUMBER);

Table created.

SQL> INSERT INTO l_tab values(1);

1 row created.

SQL> INSERT INTO l_tab values(2);
```

```
1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Connect to the CDB root as C##SEC_ADMIN to verify the status of DV_ALLOW_COMMON_OPERATION. This is the default behavior: it allows local users to create Database Vault controls on common users such as command rules.

```
$ sqlplus c##sec_admin
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 9 12:18:01 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Mon Mar 09 2020 12:16:12 +00:00
Enter password: password

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION FALSE

SQL>
```

If the status is set to TRUE, set it to FALSE with the following command:

```
SQL> EXEC DBMS_MACADM.ALLOW_COMMON_OPERATION (FALSE)

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION FALSE

SQL>
```

4. Test if the common user C##TEST1 can connect to the CDB root and to PDB20 when there is no command rule applied on the common user C##TEST1.

   a. Connect to the CDB root as C##TEST1.

```
SQL> CONNECT c##test1
Enter password: password
```

```
Connected.
SQL>
```

b. Connect to PDB20 as C##TEST1.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL>
```

5. Test if the common user C##TEST1 can connect to the CDB root and to PDB20 when there is a command rule applied on the common user C##TEST1 in the CDB root.

a. Create a command rule on C##TEST1 in the CDB root.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Disabled',
  user_name       => 'C##TEST1',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1.

```
SQL> CONNECT c##test1
Enter password: password
ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON

Warning: You are no longer connected to ORACLE.
SQL> !oerr ora 47400
47400, 00000, "Command Rule violation for %s on %s"
// *Cause: An operation that was attempted failed due to a command
rule
//         violation
// *Action: Ensure you have sufficient privileges for this
operation retry
//          the operation

SQL>
```

c. Connect to PDB20 as C##TEST1.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
```

```
Connected.
SQL>
```

d. Drop the command rule.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE('C##TEST1',DBMS_MACUTL.G_SCO
PE_LOCAL)

PL/SQL procedure successfully completed.

SQL>
```

6. Test if the common user C##TEST1 can connect to the CDB root and to PDB20 when there is a command rule applied on the common user C##TEST1 in PDB20.

a. Create a command rule on C##TEST1 in PDB20.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Disabled',
  user_name       => 'C##TEST1',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/  2    3    4    5    6    7    8

PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL>
```

c. Connect to PDB20 as C##TEST1.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON

Warning: You are no longer connected to ORACLE.

SQL>
```

d. Drop the command rule.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> EXEC
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE('C##TEST1',DBMS_MACUTL.G_SCO
PE_LOCAL)

PL/SQL procedure successfully completed.

SQL>
```

7. Connect to the CDB root as C##SEC_ADMIN and prevent local users from creating Oracle Database Vault controls on common users which would prevent them from logging in to the PDB in which their objects reside. Set DV_ALLOW_COMMON_OPERATION to TRUE.

```
SQL CONNECT c##sec_admin
Enter password: password

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION FALSE

SQL> EXEC DBMS_MACADM.ALLOW_COMMON_OPERATION

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                      STATU
------------------------- -----
DV_ALLOW_COMMON_OPERATION TRUE

SQL>
```

Note that you can execute this procedure without including any parameter to achieve a TRUE result.

8. Test if the common user C##TEST1 can connect to the CDB root and to PDB20 when there is a command rule applied on the common user C##TEST1 in the CDB root.

a. Create a command rule on C##TEST1 in the CDB root.

```
SQL> BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Disabled',
  user_name       => 'C##TEST1',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/  2    3    4    5    6    7    8
```

```
PL/SQL procedure successfully completed.

SQL>
```

b. Connect to the CDB root as C##TEST1.

```
SQL> CONNECT c##test1
Enter password: password
ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON

Warning: You are no longer connected to ORACLE.
SQL> !oerr ora 47400
47400, 00000, "Command Rule violation for %s on %s"
// *Cause: An operation that was attempted failed due to a command
rule
//         violation
// *Action: Ensure you have sufficient privileges for this
operation retry
//          the operation

SQL>
```

c. Connect to PDB20 as C##TEST1.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL>
```

d. Drop the command rule.

```
SQL> CONNECT c##sec_admin
Enter password: password
Connected.
SQL> EXEC
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE('C##TEST1',DBMS_MACUTL.G_SCO
PE_LOCAL)

PL/SQL procedure successfully completed.

SQL>
```

9. Test if the common user C##TEST1 can connect to the CDB root and to PDB20 when there is a command rule applied on the common user C##TEST1 in PDB20.

a. Create a command rule on C##TEST1 in PDB20.

```
SQL> CONNECT sec_admin@PDB20
Enter password: password
Connected.
SQL> BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Disabled',
```

```
  user_name       => 'C##TEST1',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/  2    3    4    5    6    7    8
BEGIN
*
ERROR at line 1:
ORA-47110: cannot create command rules for C##TEST1.%
ORA-06512: at "DVSYS.DBMS_MACADM", line 1872
ORA-06512: at "DVSYS.DBMS_MACADM", line 2263
ORA-06512: at line 2

SQL> !oerr ORA 47110
47110, 00000, "cannot create command rules for %s.%s"
// *Cause: When ALLOW COMMON OPERATION was set to TRUE, a smaller
scope user was not allowed to create command rules on a larger
scope user's object.
// *Action: When ALLOW COMMON OPERATION is TRUE, do not create
command rules on a larger scope user's object.

SQL>
```

b. Connect to the CDB root as C##TEST1.

```
SQL> CONNECT c##test1
Enter password: password
Connected.
SQL>
```

c. Connect to PDB20 as C##TEST1.

```
SQL> CONNECT c##test1@PDB20
Enter password: password
Connected.
SQL> EXIT
$
```

10. Disable Database Vault in both the PDB and the CDB root.

```
$ /home/oracle/labs/M104781GC10/disable_DV.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:25:59 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Mon Apr 06 2020 15:23:56 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.
```

```
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:00 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Mon Apr 06 2020 15:23:58 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:02 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Apr 6 15:26:23 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.
```

```
Total System Global Area 6442447392 bytes
Fixed Size                  9581088 bytes
Variable Size            1090519040 bytes
Database Buffers         5318377472 bytes
Redo Buffers               23969792 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

Database Vault does not only block inappropriate command rules from being created once `DBMS_MACADM.ALLOW_COMMON_OPERATION` is set to `TRUE`, but existing local command rules created when `DBMS_MACADM.ALLOW_COMMON_OPERATION` was set to `FALSE` fall under the control. Existing local command rules still exist but enforcement is ignored.

Performance and High-Availability Options

- Automatic Operations
- Oracle Database In-Memory
- Flashback
- Autonomous Health Framework
- Oracle Multitenant

Automatic Operations

- SecureFiles Defragmentation
- Automatic Index Optimization
- Automatic Zone Maps

SecureFiles Defragmentation

SecureFiles defragmentation provides online defragmentation of allocated and freed space in SecureFiles segments, for all types of SecureFiles LOBs - compressed, deduplicated, encrypted. Defragmentation can be done automatically by a background process, and the segment advisor can estimate the fragmentation levels and how much space can be saved. Defragmentation can be done mostly in-place, with some temp segment space needed to hold intermediate results.

SecureFiles defragmentation provides a transparent way to defragment or shrink the space used by SecureFiles segments, helping to reclaim space and improve performance, without compromising concurrent access to SecureFiles data, and without a significant impact on performance.

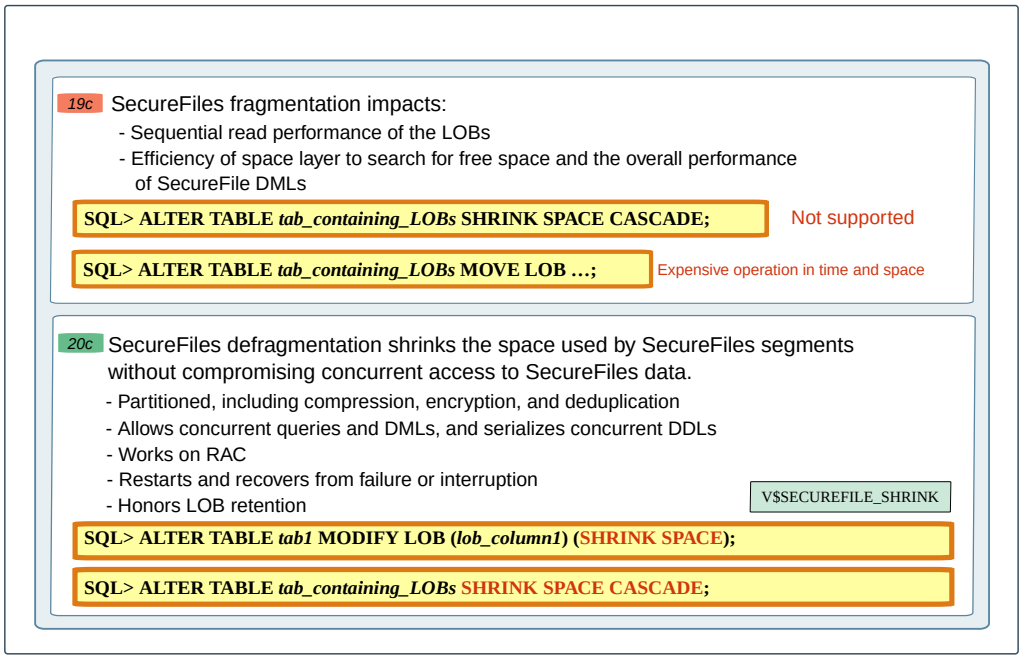
- Details: SecureFiles Defragmentation
  This page provides more detailed information about defragment operations on SecureFiles.
- Practice: Shrinking SecureFile LOBs
  This practice shows how to reclaim space and improve performance with SecureFile LOBs.

Related Topics

- *Oracle® Database SecureFiles and Large Objects Developer's Guide*

Details: SecureFiles Defragmentation

This page provides more detailed information about defragment operations on SecureFiles.



A new view `V$SECUREFILE_SHRINK` reports the results of the defragment operations. A new row is created after each invocation of shrink and is continuously updated. After the shrink is done, the row remains static, and a new invocation of shrink for the same segment overwrites the row.

Practice: Shrinking SecureFile LOBs

This practice shows how to reclaim space and improve performance with SecureFile LOBs.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Create a table with a CLOB column.

a. Before starting shrinking a SecureFile LOB, execute the `/home/oracle/labs/M104780GC10/setup_LOB.sh` shell script that creates a tablespace with sufficient space to let the LOB grow and be candidate for shrinking.

```
$ cd /home/oracle/labs/M104780GC10
$ /home/oracle/labs/M104780GC10/setup_LOB.sh

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Dec 13 11:05:28
2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

SQL> CREATE TABLESPACE users DATAFILE '/home/oracle/labs/
users01.dbf' SIZE 500M;

Tablespace created.

SQL> create user hr identified by password;

User created.

SQL> grant dba to hr;

Grant succeeded.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

b. Create a table with a CLOB column in `PDB20`.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Dec 13 11:09:44
2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Fri Dec 13 2019 10:42:50 +00:00

Connected to:
```

```
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> CREATE TABLE hr.t1 ( a CLOB) LOB(a) STORE AS SECUREFILE
TABLESPACE users;

Table created.

SQL>
```

c. Insert rows, update the CLOB data and commit.

```
SQL> INSERT INTO hr.t1 values
('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');

1 row created.

SQL> INSERT INTO hr.t1 Select * from hr.t1;

1 row created.

SQL> INSERT INTO hr.t1 Select * from hr.t1;

2 rows created.

SQL> INSERT INTO hr.t1 Select * from hr.t1;

4 rows created.

SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a||a;

8 rows updated.

SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a||a;

8 rows updated.

SQL> COMMIT;

Commit complete.

SQL>
```

3. Shrink the LOB segment.

```
SQL> ALTER TABLE hr.t1 MODIFY LOB(a) (SHRINK SPACE);

Table altered.

SQL>
```

4. Display the number of extents or blocks freed.

```
SQL> SET PAGES 100
SQL> SELECT * FROM v$securefile_shrink;

  LOB_OBJD SHRINK_STATUS
---------- ---------------------------------------
START_TIME
---------------------------------------------------------------------------
---
END_TIME
---------------------------------------------------------------------------
---
BLOCKS_MOVED BLOCKS_FREED BLOCKS_ALLOCATED EXTENTS_ALLOCATED
EXTENTS_FREED
------------ ------------ --------------- ----------------
-------------
EXTENTS_SEALED     CON_ID
-------------- ----------
     74403 COMPLETE
13-DEC-19 11.14.30.702 AM +00:00
13-DEC-19 11.14.33.520 AM +00:00
           2            2                2
1             1
              1          4


SQL>
```

As a result, two blocks are freed.

5. Update the CLOB.

```
SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a;

8 rows updated.

SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a;

8 rows updated.

SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a;

8 rows updated.

SQL> UPDATE hr.t1 SET a=a||a||a||a||a||a;

8 rows updated.

SQL> COMMIT;

Commit complete.

SQL>
```

6. Shrink the LOB segment.

```
SQL> ALTER TABLE hr.t1 MODIFY LOB(a) (SHRINK SPACE);

Table altered.

SQL>
```

7. Display the number of extents or blocks freed.

```
SQL> SELECT * FROM v$securefile_shrink;

  LOB_OBJD SHRINK_STATUS
---------- ----------------------------------------
START_TIME
---------------------------------------------------------------------------
---
END_TIME
---------------------------------------------------------------------------
---
BLOCKS_MOVED BLOCKS_FREED BLOCKS_ALLOCATED EXTENTS_ALLOCATED
EXTENTS_FREED
------------ ------------ --------------- ----------------
-------------
EXTENTS_SEALED     CON_ID
-------------- ----------
     74403 COMPLETE
13-DEC-19 11.22.07.225 AM +00:00
13-DEC-19 11.22.18.281 AM +00:00
        2648         2648             2648                 0
11
            11          4

     74403 COMPLETE
13-DEC-19 11.14.30.702 AM +00:00
13-DEC-19 11.14.33.520 AM +00:00
           2            2                2
1             1
              1          4

SQL>
```

As a result, 2648 blocks are freed. Observe that the first row remains static.

8. Update the CLOB.

```
SQL> UPDATE hr.t1 SET a=a||a;

8 rows updated.

SQL> COMMIT;

Commit complete.
```

```
SQL>
```

9. Shrink the LOB segment.

```
SQL> ALTER TABLE hr.t1 MODIFY LOB(a) (SHRINK SPACE);

Table altered.

SQL>
```

10. Display the number of extents or blocks freed.

```
SQL> SELECT * FROM v$securefile_shrink WHERE LOB_OBJD=74403;

  LOB_OBJD SHRINK_STATUS
---------- ------------------------------------
START_TIME
---------------------------------------------------------------------------
---
END_TIME
---------------------------------------------------------------------------
---
BLOCKS_MOVED BLOCKS_FREED BLOCKS_ALLOCATED EXTENTS_ALLOCATED
EXTENTS_FREED
------------ ------------ ---------------- -----------------
-------------
EXTENTS_SEALED     CON_ID
-------------- ----------
     74403 COMPLETE
13-DEC-19 11.22.07.225 AM +00:00
13-DEC-19 11.22.18.281 AM +00:00
        2648         2648             2648                 0
11
            11          4

     74403 COMPLETE
13-DEC-19 11.24.14.623 AM +00:00
13-DEC-19 11.24.39.373 AM +00:00
        5484         5484             5484                 1
19
            19          4

SQL> EXIT
$
```

As a result, 5484 blocks are freed. Observe that only the row of the previous shrinking operation is kept.

Automatic Index Optimization

ADO Policies for Indexes extends existing Automatic Data Optimization (ADO) functionality to provide compression and optimization capability on indexes. Customers of Oracle Database are interested in leveraging compression tiering and

storage tiering to satisfy their Information Lifecycle Management (ILM) requirements. Existing ADO functionality enables you to set policies that enforce compression tiering and storage tiering for data tables and partitions automatically, with minimal user intervention.

In a database, indexes can contribute to a significant amount of database space. Reducing the space requirement for indexes, without sacrificing performance, requires ILM actions similar to the existing Automatic Data Optimization feature for data segments. Using this new Index compression and optimization capability, the same ADO infrastructure can also automatically optimize indexes. Similar to ADO for data segments, this automatic index compression and optimization capability achieves ILM on indexes by enabling you to set policies that automatically optimize indexes through actions like compressing, shrinking and rebuilding indexes.

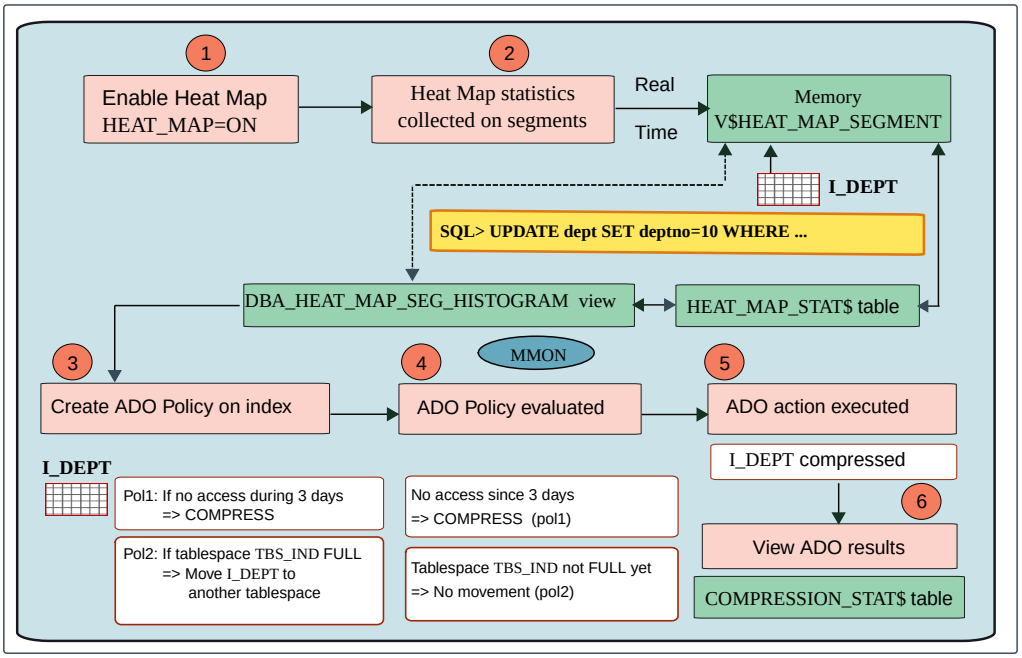
- Details: Automatic Index Optimization
  This page provides more detailed information about Automatic Data Optimization policies for indexes, extending existing ADO functionality for tables to provide segment movement, compression and optimization capability on indexes.
- Practice: Implementing Storage Tiering ADO Policy for Indexes
  This practice shows how to automate the movement of indexes to another tablespace depending on certain conditions defined in Automatic Data Optimization policies.
- Practice: Implementing Optimize ADO Policy for Indexes
  This practice shows how to automate the compression and optimization of indexes, using the existing Automatic Data Optimization (ADO) framework, depending on certain conditions defined in Automatic Data Optimization policies.

Related Topics

- *Oracle® Database VLDB and Partitioning Guide*

Details: Automatic Index Optimization

This page provides more detailed information about Automatic Data Optimization policies for indexes, extending existing ADO functionality for tables to provide segment movement, compression and optimization capability on indexes.



The slide shows how to set up the different steps between Heat Map and Automatic Data Optimization (ADO) to automate the movement of a segment to another tablespace and/or the compression of blocks or a segment depending on certain conditions defined in ADO policies.

Oracle Database 20c allows ADO policies for indexes, extending existing Automatic Data Optimization (ADO) functionality for tables to provide segment movement, compression and optimization capability on indexes. The optimization process includes actions such as compressing, shrinking, or rebuilding indexes. When the `OPTIMIZE` clause is specified, Oracle automatically determines which action is optimal for the index and implements that action as part of the optimization process. You do not have to specify which action is taken.

1. The first operation for the DBA is to enable Heat Map, tracking the activity on blocks and segments. Heat Map activates system-generated statistics collection, such as segment access or modification.

2. Real-time statistics are collected in memory (`V$HEAT_MAP_SEGMENT` view) and regularly flushed by scheduled `DBMS_SCHEDULER` jobs to the persistent `HEAT_MAP_STAT$` table. The persistent data is visible by using the `DBA_HEAT_MAP_SEG_HISTOGRAM` view.

3. The next operation for the DBA is to create ADO policies on indexes as default ADO behavior on tablespaces.

4. The next step for the DBA is to schedule when ADO policy evaluation must happen if the default scheduling does not match the business requirements. ADO policy evaluation relies on Heat Map statistics. MMON evaluates row-level policies periodically and start jobs to compress whichever blocks qualify. Segment-level policies are evaluated and executed only during the maintenance window.

5. The DBA can finally view ADO execution results by using the `DBA_ILMEVALUATIONDETAILS` and `DBA_ILMRESULTS` views.

6. Finally, the DBA can verify whether the segment moved to another tablespace and is therefore stored on the tablespace defined in the ADO policy, and or if blocks of the index got compressed viewing the `COMPRESSION_STAT$` table.

Practice: Implementing Storage Tiering ADO Policy for Indexes

This practice shows how to automate the movement of indexes to another tablespace depending on certain conditions defined in Automatic Data Optimization policies.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before creating the storage tier ADO policy on an index, execute the `/home/oracle/labs/M104783GC10/ADO_setup.sh`. The shell script cleans up any existing ADO policies, creates two tablespaces for moving indexes from the `ADOTBSINDX` tablespace to the `LOW_COST_STORE_INDX` tablespace, and creates the `HR.EMP` table with a primary key `PK_EMPLOYEE_ID` whose index is stored in the `ADOTBSINDX`. It also starts collecting the heat map statistics.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/ADO_setup.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:31:27 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> set feedback off
SQL> delete ilm_results$;
SQL> delete ilm_execution$;
SQL> delete ilm_executiondetails$;
SQL> DROP TABLESPACE adotbsindx INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE adotbsindx INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-00959: tablespace 'ADOTBSINDX' does not exist


SQL> DROP TABLESPACE low_cost_store_indx INCLUDING CONTENTS AND
DATAFILES;
DROP TABLESPACE low_cost_store_indx INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-00959: tablespace 'LOW_COST_STORE_INDX' does not exist


SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,85);
  4  exception
  5  when others then
  6  raise;
  7  end;
```

```
  8  /
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,25);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:31:28 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> set feedback off
SQL> delete ilm_results$;
SQL> delete ilm_execution$;
SQL> delete ilm_executiondetails$;
SQL> DROP TABLESPACE adotbsindx INCLUDING CONTENTS AND DATAFILES;
SQL> DROP TABLESPACE low_cost_store_indx INCLUDING CONTENTS AND
DATAFILES;
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,85);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,25);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
```

```
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:31:34 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SYSTEM SET heat_map=on SCOPE=BOTH;

System altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 6 03:29:05 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0


specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:


PL/SQL procedure successfully completed.


User created.


User altered.


User altered.


Grant succeeded.
```

```
Grant succeeded.


Session altered.


Session altered.


Session altered.

******  Creating REGIONS table ....
...
******  Creating EMPLOYEES table ....
...
******  Populating EMPLOYEES table ....
...
1 row created.
...
Index created.
...
Trigger altered.
...
PL/SQL procedure successfully completed.

SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 6 03:29:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL>
SQL> GRANT select any dictionary TO hr;

Grant succeeded.

SQL> CREATE TABLESPACE adotbsindx
  2         DATAFILE  '/home/oracle/labs/adotbs1.dbf'
  3         size 2m reuse autoextend off extent management local
uniform size 64K;

Tablespace created.

SQL> CREATE TABLESPACE low_cost_store_indx
  2         DATAFILE  '/home/oracle/labs/lcs.dbf'
  3         size 100M;
```

```
Tablespace created.

SQL>
SQL> CREATE TABLE hr.emp TABLESPACE users AS SELECT * FROM
hr.employees ;

Table created.

SQL> ALTER TABLE hr.emp MODIFY employee_id NUMBER(38) ;

Table created.

SQL> ALTER TABLE hr.emp ADD CONSTRAINT pk_employee_id primary key
(employee_id) using index tablespace adotbsindx;

Table altered.

SQL> INSERT INTO hr.emp
  2      SELECT employee_id*3, first_name,last_name, email,
phone_number, hire_date, job_id, salary, commission_pct, manager_id,
department_id
  3          FROM hr.emp;

107 rows created.

SQL> INSERT INTO hr.emp
  2      SELECT employee_id*7, first_name,last_name, email,
phone_number, hire_date, job_id, salary, commission_pct, manager_id,
department_id
  3          FROM hr.emp;

214 rows created.

SQL> COMMIT;

Commit complete.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Display the tablespace on which the index of the primary key for the `HR.EMP` table is stored and how much space the segment is using.

```
$ sqlplus system@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 6 03:36:57 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password
```

```
Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> COL tablespace_name FORMAT A20
SQL> COL index_name FORMAT A20
SQL> COL owner FORMAT A10
SQL> SELECT tablespace_name, index_name, owner FROM dba_indexes WHERE
table_name='EMP';

TABLESPACE_NAME      INDEX_NAME           OWNER
-------------------- -------------------- ----------
ADOTBSINDX           PK_EMPLOYEE_ID       HR

SQL>
SQL> SELECT bytes FROM dba_segments WHERE segment_name='PK_EMPLOYEE_ID';

     BYTES
----------
     65536

SQL>
```

4. Display the space used and free in the tablespace on which the index of the primary key for the `HR.EMP` table is stored.

```
SQL> SELECT /* + RULE */ df.tablespace_name "Tablespace",
            df.bytes / (1024 * 1024) "Size (MB)",
            SUM(fs.bytes) / (1024 * 1024) "Free (MB)",
            Nvl(Round(SUM(fs.bytes) * 100 / df.bytes),1) "% Free",
            Round((df.bytes - SUM(fs.bytes)) * 100 / df.bytes) "% Used"
     FROM dba_free_space fs, (SELECT tablespace_name,SUM(bytes) bytes
                              FROM dba_data_files
                              GROUP BY tablespace_name) df
     WHERE fs.tablespace_name (+) = df.tablespace_name
     GROUP BY df.tablespace_name,df.bytes
     ORDER BY 4;

Tablespace                     Size (MB)  Free (MB)     % Free     %
Used
------------------------------ ---------- ---------- ----------
----------
SYSTEM                                270      6.125          2
98
SYSAUX                                340    17.1875          5
95
USERS                                   5     2.3125         46
54
ADOTBSINDX                              2      .9375         47
53
UNDOTBS1                              100     66.125         66
34
LOW_COST_STORE_INDX                   100         99
99          1
```

```
6 rows selected.

SQL>
```

5. Create a storage tiering ADO policy on the index so that when the percentage of empty space in `ADOTBSINDX` tablespace is less than 90%, the ILM policy being evaluated triggers an ADO action to move the index to the `LOW_COST_STORE_INDX` tablespace.

```
SQL> ALTER INDEX hr.pk_employee_id ILM ADD POLICY TIER TO
low_cost_store_indx;

Index altered.

SQL>
```

6. Display the policy in the data dictionary view.

```
SQL> CONNECT hr@PDB20
Enter password: password
Connected.
SQL> SELECT  policy_name, action_type, scope,
             tier_tablespace "TIER_TBS"
     FROM    user_ilmdatamovementpolicies
     ORDER BY policy_name;

POLI ACTION_TYPE SCOPE   TIER_TBS
---- ----------- ------- --------------------
P61  STORAGE     SEGMENT LOW_COST_STORE_INDX

SQL>
```

7. Insert rows into `HR.EMP` until the index entries inserted raise the percentage of empty space in `ADOTBSINDX` tablespace to less than 90%.

```
SQL> INSERT INTO hr.emp
              SELECT employee_id*101, first_name,last_name, email,
                     phone_number, hire_date, job_id, salary,
commission_pct,
                     manager_id, department_id
           FROM hr.emp;

428 rows created.

SQL> INSERT INTO hr.emp
              SELECT employee_id+436926 , first_name,last_name, email,
                     phone_number, hire_date, job_id, salary,
commission_pct,
                     manager_id, department_id
              FROM hr.emp;

856 rows created.

SQL> COMMIT;
```

```
Commit complete.

SQL> SELECT /* + RULE */ df.tablespace_name "Tablespace",
            df.bytes / (1024 * 1024) "Size (MB)",
            SUM(fs.bytes) / (1024 * 1024) "Free (MB)",
            Nvl(Round(SUM(fs.bytes) * 100 / df.bytes),1) "% Free",
            Round((df.bytes - SUM(fs.bytes)) * 100 / df.bytes) "% Used"
     FROM dba_free_space fs, (SELECT tablespace_name,SUM(bytes) bytes
                              FROM dba_data_files
                              GROUP BY tablespace_name) df
     WHERE fs.tablespace_name (+) = df.tablespace_name
     GROUP BY df.tablespace_name,df.bytes
     ORDER BY 4;

Tablespace                     Size (MB)  Free (MB)     % Free     %
Used
------------------------------ ---------- ---------- ----------
----------
SYSTEM                                270      6.125          2
98
SYSAUX                                340     16.625          5
95
ADOTBSINDX                              2       .875         44
56
USERS                                   5       2.25         45
55
UNDOTBS1                              100    66.6875         67
33
LOW_COST_STORE_INDX                   100         99
99          1

6 rows selected.

SQL>
```

The index entries inserted raise the percentage of empty space in ADOTBSINDX tablespace to less than 90%.

8. Display the tablespace on which the index of the primary key for the HR.EMP table is now stored. Is the index moved to the LOW_COST_STORE_INDX tablespace?

```
SQL> SELECT tablespace_name, index_name, owner FROM dba_indexes WHERE
table_name='EMP';

TABLESPACE_NAME      INDEX_NAME           OWNER
-------------------- -------------------- ----------
ADOTBSINDX           PK_EMPLOYEE_ID       HR

SQL>
```

The index has not moved to the other tablespace although the percentage of empty space in ADOTBSINDX tablespace to less than 90%.

9. The ADO decision to move segments also depends on the default thresholds defined at the database level for all user-defined tablespaces.

a. Set the TBS_PERCENT_FREE threshold to 90% and the TBS_PERCENT_USED threshold to 30% .

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password
Connected.
SQL> COL name FORMAT A40
SQL> SELECT * FROM dba_ilmparameters;

NAME                                          VALUE
---------------------------------------- ----------
ENABLED                                           1
RETENTION TIME                                   30
JOB LIMIT                                         2
EXECUTION MODE                                    2
EXECUTION INTERVAL                               15
TBS PERCENT USED                                 85
TBS PERCENT FREE                                 25
POLICY TIME                                       0

8 rows selected.

SQL> EXEC
dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,90)

PL/SQL procedure successfully completed.

SQL> EXEC
dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,30)

PL/SQL procedure successfully completed.

SQL> SELECT * FROM dba_ilmparameters;

NAME                                          VALUE
---------------------------------------- ----------
ENABLED                                           1
RETENTION TIME                                   30
JOB LIMIT                                         2
EXECUTION MODE                                    2
EXECUTION INTERVAL                               15
TBS PERCENT USED                                 30
TBS PERCENT FREE                                 90
POLICY TIME                                       0

8 rows selected.

SQL>
```

b. For the purpose of the demo, you will not wait for the maintenance window to open to trigger the ADO policies jobs. Instead, you are going to execute the

following command that uses the following PL/SQL block connected as the ADO policy owner, HR.

```
SQL> CONNECT hr@PDB20
Enter password: password
Connected.
SQL> ALTER SESSION SET nls_date_format='dd-mon-yy hh:mi:ss';

Session altered.

SQL> DECLARE
     v_executionid number;
     BEGIN
       dbms_ilm.execute_ILM (ILM_SCOPE => dbms_ilm.SCOPE_SCHEMA,
                             execution_mode =>
dbms_ilm.ilm_execution_offline,
                             task_id   => v_executionid);
     END;
/

PL/SQL procedure successfully completed.

SQL>
```

c. Check again whether the index has moved to the LOW_COST_STORE_INDX tablespace.

```
SQL> COL object_type FORMAT A10
SQL> COL object_name FORMAT A14
SQL> COL selected_for_execution FORMAT A28
SQL> COL job_name FORMAT A9
SQL> SELECT OBJECT_TYPE, OBJECT_NAME, SELECTED_FOR_EXECUTION,
JOB_NAME
     FROM   user_ilmevaluationdetails;

OBJECT_TYP OBJECT_NAME    SELECTED_FOR_EXECUTION       JOB_NAME
---------- -------------- ---------------------------- ---------
INDEX      PK_EMPLOYEE_ID SELECTED FOR EXECUTION       ILMJOB124

SQL> SELECT task_id, job_name, job_state FROM user_ilmresults;

TASK_ID JOB_NAME   JOB_STATE
------- ---------- ----------------------------------
     41 ILMJOB124  COMPLETED SUCCESSFULLY

SQL>
```

10. Display the tablespace on which the index of the primary key for the HR.EMP table is now stored. Is it moved to the LOW_COST_STORE_INDX tablespace?

```
SQL> SELECT tablespace_name, index_name, owner FROM dba_indexes WHERE
table_name='EMP';

TABLESPACE_NAME      INDEX_NAME           OWNER
-------------------- -------------------- ----------
```

```
LOW_COST_STORE_INDX  PK_EMPLOYEE_ID       HR

SQL>
```

The index has moved to the other tablespace.

11. Delete the ADO policy on the index.

```
SQL> ALTER INDEX pk_employee_id ILM DELETE POLICY p61;

Index altered.

SQL>
```

12. Stop heat map statistics collection and clean up all heat map statistics.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET heat_map=off SCOPE=BOTH;

System altered.

SQL> EXEC dbms_ilm_admin.clear_heat_map_all

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

Practice: Implementing Optimize ADO Policy for Indexes

This practice shows how to automate the compression and optimization of indexes, using the existing Automatic Data Optimization (ADO) framework, depending on certain conditions defined in Automatic Data Optimization policies.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before creating the optimize ADO policy on an index, execute the `/home/oracle/labs/M104783GC10/ADO_setup2.sh`. The shell script cleans up any existing ADO policies, creates the `HR.EMP` table and starts collecting the heat map statistics.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/ADO_setup2.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:35:49 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> set feedback off
```

```
SQL> delete ilm_results$;
SQL> delete ilm_execution$;
SQL> delete ilm_executiondetails$;
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,85);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,25);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:35:50 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> set feedback off
SQL> delete ilm_results$;
SQL> delete ilm_execution$;
SQL> delete ilm_executiondetails$;
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,85);
  4  exception
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> declare
  2  begin
  3  dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,25);
  4  exception
```

```
  5  when others then
  6  raise;
  7  end;
  8  /
SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Jan 7 03:35:51 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SYSTEM SET heat_map=on SCOPE=BOTH;

System altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 6 03:29:05 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:

PL/SQL procedure successfully completed.

User created.

User altered.

User altered.

Grant succeeded.
```

```
Grant succeeded.

Session altered.

Session altered.

Session altered.

******  Creating REGIONS table ....
...
******  Creating EMPLOYEES table ....
...
******  Populating EMPLOYEES table ....
...
1 row created.
...
Index created.
...
Trigger altered.
...
PL/SQL procedure successfully completed.

SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 6 03:29:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL>
SQL> GRANT select any dictionary TO hr;

Grant succeeded.
SQL> DROP TABLESPACE low_cost_store_indx INCLUDING CONTENTS AND
DATAFILES;

Tablespace dropped.

SQL> CREATE TABLESPACE low_cost_store_indx
  2         DATAFILE  '/home/oracle/labs/lcs.dbf'
  3         size 100M;

Tablespace created.

SQL>
SQL> CREATE TABLE hr.emp TABLESPACE users AS SELECT * FROM
hr.employees ;
```

```
Table created.

SQL> ALTER TABLE hr.emp MODIFY employee_id NUMBER(38) ;

Table created.

SQL> ALTER TABLE hr.emp ADD CONSTRAINT pk_employee_id primary key
(employee_id) using index tablespace adotbsindx;

Table altered.

SQL> INSERT INTO hr.emp
  2      SELECT employee_id*3, first_name,last_name, email,
phone_number, hire_date, job_id, salary, commission_pct, manager_id,
department_id
  3          FROM hr.emp;

107 rows created.

SQL> INSERT INTO hr.emp
  2      SELECT employee_id*7, first_name,last_name, email,
phone_number, hire_date, job_id, salary, commission_pct, manager_id,
department_id
  3          FROM hr.emp;

214 rows created.

SQL> COMMIT;

Commit complete.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Before creating the optimization ADO policy on an index, create the composite index `I_NAME` on the two columns `FIRST_NAME` and `LAST_NAME` of the `HR.EMP` table.

```
$ sqlplus hr@PDB20
Enter password: password
Connected.
SQL> CREATE INDEX hr.i_name ON hr.emp (first_name, last_name)
TABLESPACE low_cost_store_indx;

Index created.

SQL>
```

4. Check the compression attribute of the index.

```
SQL> COL index_name FORMAT A26
SQL> SELECT compression, index_name FROM dba_indexes WHERE
table_name='EMP';
```

```
COMPRESSION   INDEX_NAME
------------- --------------------
DISABLED      PK_EMPLOYEE_ID
DISABLED      I_NAME

SQL>
```

5. Add an OPTIMIZE ADO policy on the `I_NAME` index. When the `OPTIMIZE` clause is specified, Oracle automatically determines which action is optimal for the index and implements that action as part of the optimization process. You do not have to specify which action is taken. The optimization process includes actions such as compressing, shrinking, or rebuilding indexes. The `OPTIMIZE` clause provides an opportunity for ADO to optimize the index whenever the policy condition is met. The exact action invoked by ADO would be based on the decision made by the Oracle Database. For example, if more than 30% of the leaf blocks are suitable for `COALESCE`, then `REBUILD ONLINE` may take less elapsed time and certainly generate a lot less undo.

```
SQL> ALTER INDEX hr.i_name ILM ADD POLICY OPTIMIZE AFTER 10 DAYS OF NO
MODIFICATION;

Index altered.

SQL>
```

6. Verify that the policy is added.

```
SQL> SELECT  policy_name, action_type, scope,
             compression_level, condition_type, condition_days
     FROM    user_ilmdatamovementpolicies
     ORDER BY policy_name;
  2    3    4
POLI ACTION_TYPE SCOPE   COMPRESSION_LEVEL CONDITION_TYPE
---- ----------- ------- ---------------- ---------------------
CONDITION_DAYS
--------------
P62  OPTIMIZE    SEGMENT                  LAST MODIFICATION TIME
            10

SQL>
```

7. To indicate that the policy is specified in seconds rather than in days, set the `POLICY TIME` to 1 (seconds) instead of the default value 0 (days) to test ADO policy evaluation quickly instead of waiting for the policy duration.

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password
Connected.
SQL> EXEC
dbms_ilm_admin.customize_ilm(dbms_ilm_admin.POLICY_TIME,dbms_ilm_admin.I
LM_POLICY_IN_SECONDS)

PL/SQL procedure successfully completed.
```

```
SQL> SELECT * FROM dba_ilmparameters;

NAME                 VALUE
-------------------- -----
ENABLED                  1
RETENTION TIME          30
JOB LIMIT                2
EXECUTION MODE           2
EXECUTION INTERVAL      15
TBS PERCENT USED        85
TBS PERCENT FREE        25
POLICY TIME              1

8 rows selected.

SQL>
```

8. Wait at least until 1 minute (instead of 10 days) has passed without any modification on `HR.EMP` table, and therefore on `HR.I_NAME` index. For the purpose of the demo, you will not wait until MMON evaluates the ADO policies. You launch the ADO policy evaluation and ADO task execution immediately by executing the following PL/SQL block.

```
SQL> CONNECT hr@PDB20
Enter password: password
Connected.
SQL> ALTER SESSION SET nls_date_format='dd-mon-yy hh:mi:ss';

Session altered.

SQL> DECLARE
v_executionid number;
BEGIN
dbms_ilm.execute_ILM (ILM_SCOPE => dbms_ilm.SCOPE_SCHEMA,
            execution_mode => dbms_ilm.ilm_execution_offline,
            task_id   => v_executionid);
END;
/

PL/SQL procedure successfully completed.

SQL>
```

9. Display the result of the executed task.

```
SQL> COL SELECTED_FOR_EXECUTION FORMAT A28
SQL> COL job_name FORMAT A9
SQL> SELECT task_id, task_owner, state FROM dba_ilmtasks WHERE
task_owner='HR';

TASK_ID TASK_OWN STATE
------- -------- ---------
     42 HR       COMPLETED
```

```
SQL> SELECT task_id, policy_name, object_name,
            selected_for_execution, job_name
     FROM   dba_ilmevaluationdetails
     WHERE  object_name='I_NAME';

TASK_ID POLI OBJECT_N SELECTED_FOR_EXECUTION         JOB_NAME
------- ---- -------- ------------------------------ ---------
     42 P62  I_NAME   STATISTICS NOT AVAILABLE
```

10. Gather the index statistics.

```
SQL> ANALYZE INDEX hr.i_name COMPUTE STATISTICS;

Index analyzed.

SQL>
```

11. Wait at least 1 minute before re-launching the ADO policy evaluation and ADO task execution immediately.

```
SQL> DECLARE
v_executionid number;
BEGIN
dbms_ilm.execute_ILM (ILM_SCOPE => dbms_ilm.SCOPE_SCHEMA,
            execution_mode => dbms_ilm.ilm_execution_offline,
            task_id   => v_executionid);
END;
/

PL/SQL procedure successfully completed.

SQL> SELECT task_id, task_owner, state FROM dba_ilmtasks WHERE
task_owner='HR';

TASK_ID TASK_OWN STATE
------- -------- ---------
     42 HR       COMPLETED
     43 HR       COMPLETED

SQL> SELECT task_id, policy_name, object_name,
            selected_for_execution, job_name
     FROM   dba_ilmevaluationdetails
     WHERE  object_name='I_NAME';

TASK_ID POLI OBJECT_N SELECTED_FOR_EXECUTION       JOB_NAME
------- ---- -------- ---------------------------- ---------
     43 P62  I_NAME   PRECONDITION NOT SATISFIED
     42 P62  I_NAME   STATISTICS NOT AVAILABLE

SQL>
```

If the `PRECONDITION NOT SATISFIED` does not appear, generate more entries in the index. Proceed with step 12. In all cases, proceed with step 12.

12. Generate more entries in the index by inserting more rows into the table. Use the `/home/oracle/labs/M104783GC10/ADO_loop_insert.sql` SQL script.

```
SQL> @/home/oracle/labs/M104783GC10/ADO_loop_insert.sql
SQL> SET ECHO ON
SQL> CONNECT hr/password@PDB20
Connected.

SQL> INSERT INTO hr.emp
  2      SELECT employee_id + (select max(employee_id) from hr.emp),
first_name,last_name, email, phone_number, hire_date, job_id, salary,
commission_pct, manager_id, department_id
  3          FROM hr.emp;

428 rows created.
...
SQL> INSERT INTO hr.emp
  2      SELECT employee_id + (select max(employee_id) from hr.emp),
first_name,last_name, email, phone_number, hire_date, job_id, salary,
commission_pct, manager_id, department_id
  3          FROM hr.emp;

109568 rows created.

SQL> COMMIT;

Commit complete.

SQL>
```

13. Gather the index statistics using the `ANALYZE` command.

```
SQL> ANALYZE INDEX hr.i_name COMPUTE STATISTICS;

Index analyzed.

SQL>
```

14. Wait at least 1 minute before re-launching the ADO policy evaluation and ADO task execution immediately.

```
SQL> DECLARE
v_executionid number;
BEGIN
dbms_ilm.execute_ILM (ILM_SCOPE => dbms_ilm.SCOPE_SCHEMA,
            execution_mode => dbms_ilm.ilm_execution_offline,
            task_id   => v_executionid);
END;
/

PL/SQL procedure successfully completed.

SQL> SELECT task_id, task_owner, state FROM dba_ilmtasks WHERE
task_owner='HR';
```

```
TASK_ID TASK_OWN STATE
------- -------- ---------
     42 HR       COMPLETED
     43 HR       COMPLETED
     44 HR       COMPLETED

SQL> SELECT task_id, policy_name, object_name,
            selected_for_execution, job_name
     FROM   dba_ilmevaluationdetails
     WHERE  object_name='I_NAME';

TASK_ID POLI OBJECT_N SELECTED_FOR_EXECUTION       JOB_NAME
------- ---- -------- --------------------------- ---------
     44 P62  I_NAME   SELECTED FOR EXECUTION       ILMJOB164
     43 P62  I_NAME   PRECONDITION NOT SATISFIED
     42 P62  I_NAME   STATISTICS NOT AVAILABLE

SQL>
```

In case the precondition for execution is still not satisfied, generate more entries in the index by inserting more rows into the table. Use the `/home/oracle/labs/M104783GC10/ADO_loop_insert2.sql` SQL script. Then re-execute steps 13 and 14.

15. Display the compression attribute of the index.

```
SQL> SELECT compression, index_name FROM dba_indexes WHERE
table_name='EMP';

COMPRESSION   INDEX_NAME
------------- --------------------
DISABLED      PK_EMPLOYEE_ID
ADVANCED LOW  I_NAME

SQL>
```

16. Delete the ADO policy on the index.

```
SQL> ALTER INDEX hr.i_name ILM DELETE POLICY p62;

Index altered.

SQL>
```

17. Stop heat map statistics collection and clean up all heat map statistics.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET heat_map=off SCOPE=BOTH;

System altered.

SQL> EXEC dbms_ilm_admin.clear_heat_map_all
```

```
PL/SQL procedure successfully completed.

SQL> EXIT
$
```

Automatic Zone Maps

Automatic zone maps are created and maintained for any user table without any customer intervention. Zone maps allow the pruning of blocks and partitions based on the predicates in the queries, without any user intervention. Automatic zone maps are maintained for direct loads, and are maintained and refreshed for any other DML operation incrementally and periodically in the background.

Automatic zone maps are improving the performance of any query transparently and automatically without the need of any user action.

- Details: Automatic Zone Maps
  This page provides more detailed information about the automatic zone map creation and maintenance.
- Details: Automatic Zone Maps - Package
  This page provides more detailed information about the new package related to automatic zone maps.
- Details: Automatic Zone Maps - Views
  This page provides more detailed information about the new package and views related to automatic zone maps.
- Practice: Using Automatic Zone Maps
  This practice shows how to enable automatic zone maps and how automatic zone maps are created and maintained for any user table without your intervention.

Related Topics

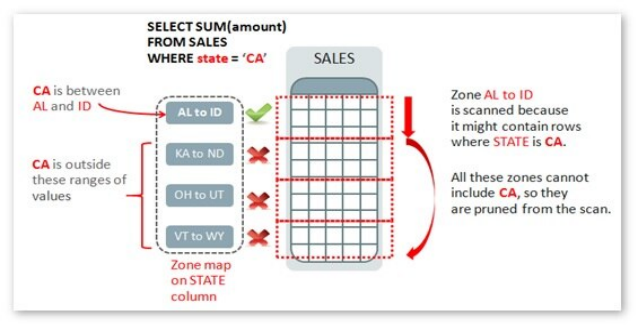
- *Oracle® Database Data Warehousing Guide*

Details: Automatic Zone Maps

This page provides more detailed information about the automatic zone map creation and maintenance.

Divide a table up into contiguous regions of blocks called zones to increase query performance

- A zone map on the STATE column records the minimum and maximum values for each zone in the table.
- A query with a WHERE clause on the STATE column skips the zones that don't contain rows for the value.



19c

- Zone maps are explicitly created and controlled by the DBA on a table-by-table basis.

20c

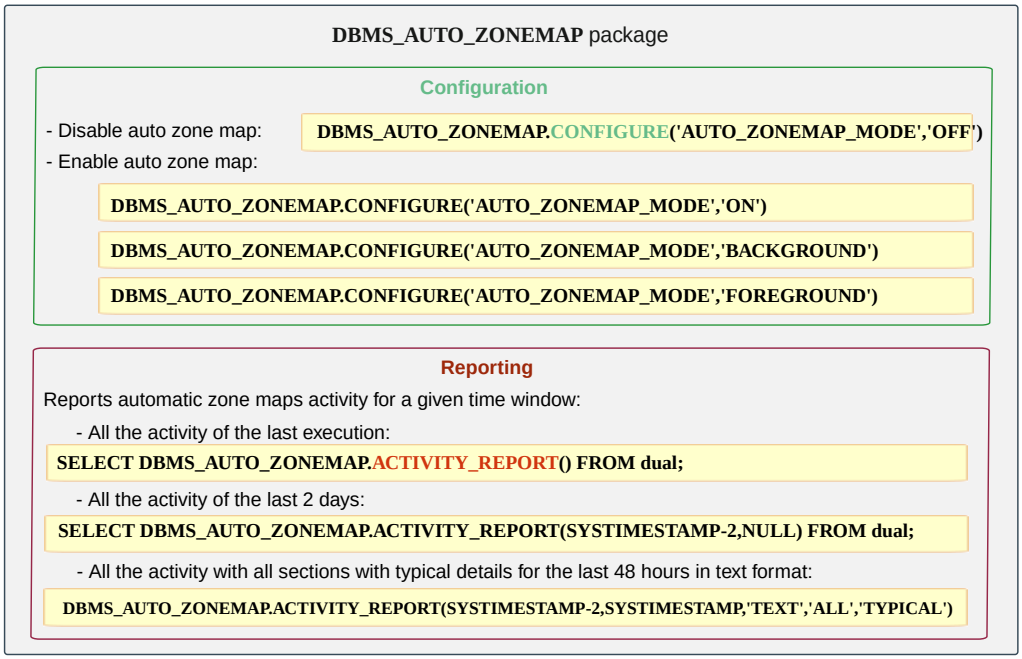
- Automatic zone maps are created and maintained for any user table without any customer intervention. Zone maps can still be created manually.

Oracle Database 20c allows you to enable automatic creation and maintenance of basic zone maps for both partitioned and non-partitioned tables by using a new package and procedure, `DBMS_AUTO_ZONEMAP.CONFIGURE`. Automatic zone map creation is turned off by default. Enabling automatic zone map creation does not require any DBA intervention any longer for both the creation of the zone maps and their maintenance. Nevertheless zone maps can still be created manually. Automatic is for Cloud autonomous database and Exadata only.

This functionality is not available for join zone maps, IOTs (Oracle Index-organized Tables), external tables, or temporary tables.

Details: Automatic Zone Maps - Package

This page provides more detailed information about the new package related to automatic zone maps.



The `DBMS_AUTO_ZONEMAP.CONFIGURE` new package and procedure allows you to set configuration options for automatic zone map, specifically to enable or disable the feature and to control foreground or background mode of the feature. There are four values allowed for the second parameter:

- `ON`: Turns on automatic zone map completely, both for foreground and background zone maps creation and maintenance
- `OFF`: Turns off automatic zone map completely, both for foreground and background zone maps creation and maintenance
- `FOREGROUND`: Turns on only for foreground zone maps creation and maintenance
- `BACKGROUND`: Turns on only for background zone maps creation and maintenance

The `DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT` reports automatic zone maps activity for a given time window. Since zone maps autotask background job is scheduled for every 15 minutes and run for one hour or less, users can query the actions performed by the zone map autotask for a given time window. The function uses four parameters:

- `START_TIME`: Timestamp from which auto zone map executions are observed for the report. NULL value reports everything from the beginning of auto zone maps maintenance. Default value is NULL.
- `END_TIME`: Timestamp until which auto zone map executions are observed for the report. NULL value reports everything to the end of auto zone maps maintenance. Default value is NULL.

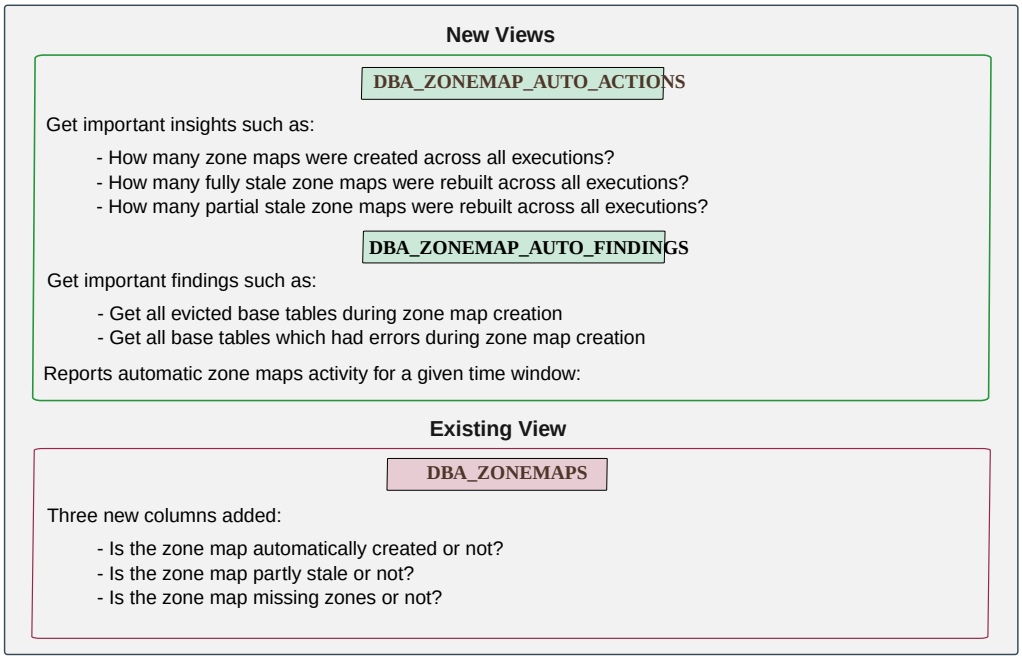
> **Note:**
>
> If NULL is specified for both `START_TIME` and `END_TIME`, `DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT` reports activity of the last execution.

- `TYPE`: Output type of the report. Possible values are: `TEXT`, `XML` and `HTML`. Default value is `TEXT`.

- `SECTION`: Particular section in the report. Possible values are `SUMMARY`, `DETAILS` and `ALL`. Default value is `ALL`.
  - `SUMMARY`: Very high level numbers summary on new zone maps created and maintained for the given time window
  - `DETAILS`: Detailed summary report on names and other details of new zone maps created and maintained for the given time window. It also includes findings details.
  - `ALL`: In addition to summary and details, it includes time series based execution / action logs.
- `LEVEL`: Format of the report. It represents the level of details with in each section. Possible values are `BASIC`, `TYPICAL` and `ALL`. Default being `TYPICAL`.
  - `BASIC`: Represents very high level details in executive summary. Users only see numbers on zone maps that were created, complete rebuilt and fast rebuilt. In new zone map details section, you can see new zone map name, date created and base table name. Maintenance details section shows only zone map name, previous state and current state. Similarly, findings section shows only object name and blacklist reason, and no other details. Action logs section shows only important time series based log messages pertaining to zone maps creation and maintenance.
  - `TYPICAL`: Everything in basic level and little more comprehensive than basic. This level shows full overview on executive summary section. New zone maps details shows schema name, column list and date created. Zone maps maintenance details section shows refresh type, date maintained. Findings section shows timestamp and exception message. Action logs section shows little more comprehensive logs than basic, which has information about candidate column list, findings information and creation DDLs.
  - `ALL`: On top of typical level are shown DOP used for each operation for creating or maintaining zone maps, time took to process each DDL and other details in action logs. Show all log messages with details on clustering ratios of columns, exception messages and other details.

Details: Automatic Zone Maps - Views

This page provides more detailed information about the new package and views related to automatic zone maps.



The `DBA_ZONEMAP_AUTO_ACTIONS` new view holds five columns:

- `TASK_ID`: Advisor task id for automatic zone maps
- `MSG_ID`: Message ID
- `EXEC_NAME`: Advisor execution name: `SYS_ZMAP_<Timestamp>`
- `ACTION_MSG`: Execution message log
- `TIME_STAMP`: Message time stamp

The `DBA_ZONEMAP_AUTO_FINDINGS` new view holds five columns:

- `TASK_ID`: Advisor task id for automatic zone maps
- `MSG_ID`: Message ID
- `EXEC_NAME`: Advisor execution name: `SYS_ZMAP_<Timestamp>`
- `MESSAGE`: Execution message log
- `TIME_STAMP`: Message time stamp
- `OBJECT_NAME`: Object name, typically table name or zone map name on which the finding was observed
- `FINDING_REASON`: Finding reason can be an error, an eviction or a timed out.
- `FINDING_TYPE`: Finding type can be a blacklist, back in queue and others.

The exsiting `DBA_ZONEMAPS` view holds three new columns:

- `AUTOMATIC`: Is the zone map automatically created or not?
- `PARTLY_STALE`: Is the zone map partly stale or not?
- `INCOMPLETE`: Is the zone map missing zones or not?

Practice: Using Automatic Zone Maps

This practice shows how to enable automatic zone maps and how automatic zone maps are created and maintained for any user table without your intervention.

1. Before starting any new practice, refer to the practices environment recommendations.
2. The first part of the practice is to show how zone maps are created and visible under DBA intervention.
   a. Use the `/home/oracle/labs/M104784GC10/setup_zonemap.sh` shell script to create the `SALES.ZM_TABLE` table in `PDB20`.

```
$ cd /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/setup_zonemap.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 25 10:37:32
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0


System altered.

Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 25 10:37:33
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
```

```
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 25 10:37:58
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area 1426062424 bytes
Fixed Size                  9567320 bytes
Variable Size             855638016 bytes
Database Buffers          553648128 bytes
Redo Buffers                7208960 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 25 10:38:32
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> drop user sales cascade;
drop user sales cascade
          *
ERROR at line 1:
ORA-01918: user 'SALES' does not exist


SQL> create user sales identified by password;

User created.

SQL> grant create session, create table, unlimited tablespace to
sales;

Grant succeeded.
```

```
SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Feb 25 10:38:33
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> CREATE TABLE sales_zm (sale_id NUMBER(10), customer_id
NUMBER(10));

Table created.

SQL>
SQL>  DECLARE
  2    i NUMBER(10);
  3  BEGIN
  4    FOR i IN 1..80
  5    LOOP
  6      INSERT INTO sales_zm
  7      SELECT ROWNUM, MOD(ROWNUM,1000)
  8      FROM   dual
  9      CONNECT BY LEVEL <= 100000;
 10      COMMIT;
 11    END LOOP;
 12  END;
 13  /

PL/SQL procedure successfully completed.

SQL>
SQL> EXEC dbms_stats.gather_table_stats(ownname=>NULL,
tabname=>'SALES_ZM')

PL/SQL procedure successfully completed.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

b. Log in PDB20 as SALES, set your session in statistic trace, and query the SALES_ZM table a few times to see the “consistent gets” value.

```
$ sqlplus sales@PDB20

Enter password: password

SQL> SET AUTOTRACE ON STATISTIC
SQL> SELECT COUNT(DISTINCT sale_id) FROM sales_zm WHERE customer_id
= 50;

COUNT(DISTINCTSALE_ID)
----------------------
                   100

Statistics
----------------------------------------------------------
         44  recursive calls
         12  db block gets
      15248  consistent gets
          4  physical reads
       2084  redo size
        582  bytes sent via SQL*Net to client
        432  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
```

c. Create a zone map. Since attribute clustering is a property of the table, any existing rows are not re-ordered. Therefore move the table to cluster the rows together.

```
SQL> ALTER TABLE sales_zm ADD CLUSTERING BY LINEAR ORDER
(customer_id) WITH MATERIALIZED ZONEMAP;

Table altered.

SQL> ALTER TABLE sales_zm MOVE;

Table altered.

SQL>
```

d. Re-run the query to see the “consistent gets” value.

```
SQL> SELECT COUNT(DISTINCT sale_id) FROM sales_zm WHERE customer_id
= 50;

COUNT(DISTINCTSALE_ID)
----------------------
                   100
```

```
Statistics
----------------------------------------------------------
         67  recursive calls
          8  db block gets
        900  consistent gets
          0  physical reads
       1464  redo size
        582  bytes sent via SQL*Net to client
        432  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
```

e. Display the status of the zone map created for this table.

```
SQL> SET AUTOTRACE OFF
SQL> COL zonemap_name FORMAT A20
SQL> SELECT zonemap_name,automatic,partly_stale, incomplete
      FROM   dba_zonemaps;

ZONEMAP_NAME         AUTOMATIC PARTLY_STALE INCOMPLETE
-------------------- --------- ------------ ------------
ZMAP$_SALES_ZM       NO        NO           NO

SQL>
```

Note that the new column `AUTOMATIC`, added to the existing view `DBA_ZONEMAPS` shows that the zone map is not created automatically.

3. The second part of the practice is to show how to enable automatic zone maps and then verify that zone maps are automatically created and how to display automatic zone map activity (creation and maintenance).

a. Drop the table.

```
SQL> DROP TABLE sales_zm PURGE;

Table dropped.

SQL> SELECT zonemap_name, automatic, partly_stale, incomplete
     FROM   dba_zonemaps;

no rows selected

SQL>
```

b. Enable automatic zone map creation.

```
SQL> EXEC DBMS_AUTO_ZONEMAP.CONFIGURE('AUTO_ZONEMAP_MODE','ON')

PL/SQL procedure successfully completed.
```

```
SQL>
```

c. Re-create the table, insert rows with direct load, and gather table statistics.

```
SQL> CREATE TABLE sales_zm (sale_id NUMBER(10), customer_id
NUMBER(10));

Table created.

SQL> DECLARE
  i NUMBER(10);
BEGIN
  FOR i IN 1..80
  LOOP
    INSERT /*+ APPEND */ INTO sales_zm
    SELECT ROWNUM, MOD(ROWNUM,1000)
    FROM   dual
    CONNECT BY LEVEL <= 100000;
    COMMIT;
  END LOOP;
END;
/  2    3    4    5    6    7    8    9   10   11   12   13
PL/SQL procedure successfully completed.

SQL> EXEC dbms_stats.gather_table_stats(ownname=>NULL,
tabname=>'SALES_ZM')

PL/SQL procedure successfully completed.

SQL>
```

d. Query the SALES_ZM table at least twenty times to see the “consistent gets” value.

```
SQL> SET AUTOTRACE ON STATISTIC
SQL> SELECT COUNT(DISTINCT sale_id) FROM sales_zm WHERE customer_id
= 50;

COUNT(DISTINCTSALE_ID)
----------------------
                   100

Statistics
----------------------------------------------------------
         44  recursive calls
         12  db block gets
      15248  consistent gets
          4  physical reads
       2084  redo size
        582  bytes sent via SQL*Net to client
        432  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
```

```
          1  rows processed

SQL> /

COUNT(DISTINCTSALE_ID)
----------------------
                   100

Statistics
----------------------------------------------------------
         44  recursive calls
         12  db block gets
      15248  consistent gets
          4  physical reads
       2084  redo size
        582  bytes sent via SQL*Net to client
        432  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL> /

COUNT(DISTINCTSALE_ID)
----------------------
                   100

Statistics
----------------------------------------------------------
         44  recursive calls
         12  db block gets
      15248  consistent gets
          4  physical reads
       2084  redo size
        582  bytes sent via SQL*Net to client
        432  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
```

e. Because the background process responsible for the zone maps creation will wake up late, use the `/home/oracle/labs/M104784GC10/zonemap_exec.sql` SQL script to wake it up sooner.

```
SQL> @/home/oracle/labs/M104784GC10/zonemap_exec.sql
Connected.

PL/SQL procedure successfully completed.
```

```
Connected.
SQL>
```

f. Display the status of the zone map created.

```
SQL> SELECT zonemap_name, automatic, partly_stale, incomplete
     FROM   dba_zonemaps;

ZONEMAP_NAME         AUTOMATIC PARTLY_STALE INCOMPLETE
-------------------- --------- ------------ ------------
ZMAP$_SALES_ZM       YES       NO           NO

SQL>
```

g. Display the automatic zone map task actions. Query the DBA_ZONEMAP_AUTO_ACTIONS view several times until you see that an automatic zone map is created.

```
SQL> SELECT task_id, msg_id, action_msg  FROM
dba_zonemap_auto_actions;

   TASK_ID     MSG_ID
---------- ----------
ACTION_MSG
--------------------------------------------------------------------
------------
         6         35
BS:Current execution task id: 6 Execution name:
SYS_ZMAP_2020-04-06/07:56:54 Tas
k Name: ZMAP_TASK1

         6         36
BS:******** Zonemap Background Action Report for Task ID: 6
***************

         6         37
BS:****** End of Zonemap Background Action Report for Task ID: 6
**********

         6         21
BS:Current execution task id: 6 Execution name:
SYS_ZMAP_2020-04-06/07:34:36 Tas
k Name: ZMAP_TASK1

         6         22
BS:******** Zonemap Background Action Report for Task ID: 6
***************

         6         23
TP:Trying to create zonemap on table: SALES_ZM owner:SALES

         6         24
AL:Block count : 15447, sample percent is : 3.236874

         6         25
```

```
TP:col name:CUSTOMER_ID: clustering ratio: .98

         6         26
TP:col name:SALE_ID: clustering ratio: .09

         6         27
TP:Candidate column list:SALE_ID

         6         28
TP:New zonemap name: ZMAP$_SALES_ZM

         6         29
TP:Creating new zonemap ZMAP$_SALES_ZM on table SALES_ZM owner
SALEStable space
USERS

         6         30
BS:succesfully created zonemap: ZN:ZMAP$_SALES_ZM BT:SALES_ZM
SN:SALES CL:SALE_I
D CT:+00 00:00:01.605120 TS:2020-04-06/07:34:39 DP:4

         6         31
BS:****** End of Zonemap Background Action Report for Task ID: 6
**********

         6         32
BS:Current execution task id: 6 Execution name:
SYS_ZMAP_2020-04-06/07:43:46 Tas
k Name: ZMAP_TASK1

         6         33
BS:******** Zonemap Background Action Report for Task ID: 6
****************

         6         34
BS:****** End of Zonemap Background Action Report for Task ID: 6
**********


17 rows selected.

SQL>
```

Another way to show the activity report of the auto task run is to use the `DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT` function.

```
SQL> SELECT dbms_auto_zonemap.activity_report(systimestamp-2,
systimestamp, 'TEXT') FROM dual;

DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT(SYSTIMESTAMP-2,SYSTIMESTAMP,'TEXT'
)
--------------------------------------------------------------------
------------
/orarep/autozonemap/main%3flevel%3d GENERAL SUMMARY
```

```
------------------------------------------------------------------
-----------
 Activity Start    04-APR-2020 16:45:33.000000000 +00:00
 Activity End      06-APR-2020 16:45:33.656170000 +00:00
 Total Executions  1
------------------------------------------------------------------
-----------

EXECUTION SUMMARY
------------------------------------------------------------------
-----------
 zonemaps created                     1
 zonemaps compiled                    0
 zonemaps dropped                     0
 Stale zonemaps complete refreshed    0
 Partly stale zonemaps fast refreshed 0
 Incomplete zonemaps fast refreshed   0
------------------------------------------------------------------
-----------

NEW ZONEMAPS DETAILS
------------------------------------------------------------------
-----------
 Zonemap         Base Table  Schema  Operation time  Date
created         DOP  C
olumn list
  ZMAP$_SALES_ZM  SALES_ZM    SALES   00:00:01.68
2020-04-06/16:45:04  2    S
ALE_ID
------------------------------------------------------------------
-----------

ZONEMAPS MAINTENANCE DETAILS
------------------------------------------------------------------
-----------
 Zonemap  Previous State  Current State  Refresh Type  Operation
Time  Dop  Date
 Maintained
------------------------------------------------------------------
-----------

FINDINGS
------------------------------------------------------------------
-----------
 Execution Name  Finding Name  Finding Reason  Finding Type  Message


SQL>
```

If you want to know how many zone maps were created across all executions, run the following query:

```
SQL> SELECT * FROM dba_zonemap_auto_actions
WHERE action_msg LIKE '%succesfully created zonemap:%' ORDER BY
```

```
TIME_STAMP;

   TASK_ID     MSG_ID
---------- ----------
EXEC_NAME
------------------------------------------------------------------------
------------
ACTION_MSG
------------------------------------------------------------------------
------------
TIME_STAMP
------------------------------------------------------------------------
-------
         6         49
SYS_ZMAP_2020-04-06/16:45:01
BS:succesfully created zonemap: ZN:ZMAP$_SALES_ZM BT:SALES_ZM
SN:SALES CL:SALE_I
D CT:+00 00:00:01.681134 TS:2020-04-06/16:45:04 DP:2
06-APR-20 04.45.04.000000000 PM

SQL>
```

h. Update the `SALE_ID` column vales in `SALES_ZM` table. Execute the `/home/oracle/labs/M104784GC10/zonemap_update.sql` SQL script.

```
SQL> @/home/oracle/labs/M104784GC10/zonemap_update.sql
8000 rows updated.

8000 rows updated.

8000 rows updated.

8000 rows updated.

Commit complete.

SQL>
```

i. Display the status of the zone map maintenance.

```
SQL> SELECT zonemap_name, automatic, partly_stale, incomplete
     FROM   dba_zonemaps;

ZONEMAP_NAME         AUTOMATIC PARTLY_STALE INCOMPLETE
-------------------- --------- ------------ ------------
ZMAP$_SALES_ZM       YES       YES          NO

SQL>
```

j. Display the activity report until you see actions to automatic zone map maintenance.

```
SQL> SELECT dbms_auto_zonemap.activity_report(systimestamp-2,
systimestamp, 'TEXT') FROM dual;
```

```
DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT(SYSTIMESTAMP-2,SYSTIMESTAMP,'TEXT'
)
-------------------------------------------------------------------------
------------
/orarep/autozonemap/main%3flevel%3d GENERAL SUMMARY
-------------------------------------------------------------------------
-----------
 Activity Start    04-APR-2020 16:47:10.000000000 +00:00
 Activity End      06-APR-2020 16:47:10.417146000 +00:00
 Total Executions  1
-------------------------------------------------------------------------
-----------

EXECUTION SUMMARY
-------------------------------------------------------------------------
-----------
 zonemaps created                     1
 zonemaps compiled                    0
 zonemaps dropped                     0
 Stale zonemaps complete refreshed    0
 Partly stale zonemaps fast refreshed  1
 Incomplete zonemaps fast refreshed   0
-------------------------------------------------------------------------
-----------

NEW ZONEMAPS DETAILS
-------------------------------------------------------------------------
-----------
 Zonemap         Base Table  Schema  Operation time  Date
created         DOP  C
olumn list
 ZMAP$_SALES_ZM  SALES_ZM    SALES   00:00:01.60
2020-04-06/07:34:39  4    S
ALE_ID
-------------------------------------------------------------------------
-----------

ZONEMAPS MAINTENANCE DETAILS
-------------------------------------------------------------------------
-----------
 Zonemap         Previous State  Current State  Refresh Type
Operation Time  Do
p  Date Maintained
 ZMAP$_SALES_ZM  PARTLY_STALE    VALID          REBUILD
00:00:01.77     0
   2020-04-06/08:41:24
-------------------------------------------------------------------------
-----------

FINDINGS
-------------------------------------------------------------------------
-----------
 Execution Name  Finding Name  Finding Reason  Finding Type  Message
```

```
SQL>
```

It is possible that the background process responsible for the zone maps maintenance woke up very quickly and already rebuilt the zonemap. In this case, no information in "ZONEMAPS MAINTENANCE DETAILS" would be displayed.

k. It is possible that the background process responsible for the zone maps maintenance will wake up late. Use the /home/oracle/labs/M104784GC10/zonemap_exec.sql SQL script to wake it up sooner.

```
SQL> @/home/oracle/labs/M104784GC10/zonemap_exec.sql
Connected.

PL/SQL procedure successfully completed.

Connected.
SQL>
```

l. Display the activity report.

```
SQL> SELECT zonemap_name, automatic, partly_stale, incomplete
     FROM   dba_zonemaps;

ZONEMAP_NAME         AUTOMATIC PARTLY_STALE INCOMPLETE
-------------------- --------- ------------ ------------
ZMAP$_SALES_ZM       YES       NO           NO

SQL> SELECT dbms_auto_zonemap.activity_report(systimestamp-2,
systimestamp, 'TEXT') FROM dual;

DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT(SYSTIMESTAMP-2,SYSTIMESTAMP,'TEXT'
)
--------------------------------------------------------------------------
------------
/orarep/autozonemap/main%3flevel%3d GENERAL SUMMARY
--------------------------------------------------------------------------
-----------
 Activity Start    04-APR-2020 16:51:21.000000000 +00:00
 Activity End      06-APR-2020 16:51:21.228968000 +00:00
 Total Executions  2
--------------------------------------------------------------------------
-----------

EXECUTION SUMMARY
--------------------------------------------------------------------------
-----------
 zonemaps created                     1
 zonemaps compiled                    0
 zonemaps dropped                     0
 Stale zonemaps complete refreshed    0
 Partly stale zonemaps fast refreshed 1
 Incomplete zonemaps fast refreshed   0
--------------------------------------------------------------------------
-----------
```

```
NEW ZONEMAPS DETAILS
------------------------------------------------------------------------
-----------
 Zonemap         Base Table  Schema  Operation time  Date
created         DOP  C
olumn list
 ZMAP$_SALES_ZM  SALES_ZM    SALES   00:00:01.68
2020-04-06/16:45:04  2    S
ALE_ID
------------------------------------------------------------------------
-----------

ZONEMAPS MAINTENANCE DETAILS
------------------------------------------------------------------------
-----------
 Zonemap         Previous State  Current State  Refresh Type
Operation Time  Do
p  Date Maintained
 ZMAP$_SALES_ZM  PARTLY_STALE    VALID          REBUILD
00:00:05.25     0
   2020-04-06/16:48:30
------------------------------------------------------------------------
-----------

FINDINGS
------------------------------------------------------------------------
-----------
 Execution Name  Finding Name  Finding Reason  Finding Type  Message


SQL>
```

m. Drop the table.

```
SQL> DROP TABLE sales_zm PURGE;

Table dropped.

SQL> SELECT zonemap_name, automatic, partly_stale, incomplete
     FROM   dba_zonemaps;

no rows selected

SQL> EXIT
$
```

Oracle Database In-Memory

- Database In-Memory Base Level
- Automatic In-Memory
- In-Memory Hybrid Scans

- Database In-Memory External Table Enhancements

Database In-Memory Base Level

Database In-Memory is an option to Enterprise Edition. Database In-Memory now has a new "Base Level" feature. This allows the use of Database In-Memory with up to a 16GB column store without triggering any license tracking.

The feature allows you to use Database In-Memory without having to license the option. The column store is limited to 16GB when using the Base Level. This helps to show the value of Database In-Memory without having to worry about licensing issues.

Related Topics

- *Oracle® Database In-Memory Guide*

Automatic In-Memory

Automatic In-Memory enables, populates, evicts, and recompresses segments without user intervention.

When `INMEMORY_AUTOMATIC_LEVEL` is set to `HIGH`, the database automatically enables and populates segments based on their usage patterns. Combined with support for selective column level eviction and recompression, In-Memory population is largely self-managing. This automation helps maximize the number of objects that can be populated into the In-Memory Column Store at one time.

- Details: Automatic In-Memory
  This page provides more detailed information about how the new value of the initialization parameter `INMEMORY_AUTOMATIC_LEVEL` influences the behavior of in-memory segments compression in the In-Memory Column Store, population into the In-Memory Column Store and eviction from the In-Memory Column Store.
- Practice: Configuring and Observing Automatic In-Memory
  This practice shows how to configure Automatic In-Memory and then observe how in-memory objects are automatically and dynamically populated in the IM column store without user intervention, and then possibly automatically evicted from the IM column store.

Related Topics

- *Oracle® Database In-Memory Guide*

Details: Automatic In-Memory

This page provides more detailed information about how the new value of the initialization parameter `INMEMORY_AUTOMATIC_LEVEL` influences the behavior of in-memory segments compression in the In-Memory Column Store, population into the In-Memory Column Store and eviction from the In-Memory Column Store.

Automatic In-Memory optimizes the SQL workload as it changes, without manual intervention.

The working data set consists of the most frequently queried segments. Typically, the working data set changes with time for many applications. Users must decide which segments to enable as `INMEMORY`, monitor usage to decide which IM segments to populate and evict, and create ADO IM policies. These tasks require a thorough understanding of the workload.

To free the DBA from manual maintenance chores, Automatic In-Memory uses frequently updated internal statistics to maintain the working data set in the IM column store. Oracle Database decides what to populate and what to evict, and when to do it. In a sense, the IM column store becomes "self-driving."

When the initialization parameter `INMEMORY_AUTOMATIC_LEVEL` is set to `HIGH`, Automatic In-Memory continuously monitors column statistics in the IM store, and sets all segments that do not have a pre-existing `INMEMORY` attribute as `INMEMORY MEMCOMPRESS AUTO`. The database populates only objects that it decides belong in the working data set. This decision is based on current usage statistics. The database identifies cold regions of the IM store through internal column statistics, which are similar to those used by Heat Map but do not require `HEAT_MAP` to be set to `ON`. Automatic In-Memory can recompress cold columns in `AUTO` segments to save space. Segments with a `PRIORITY` setting other than `NONE` are excluded from the automatic eviction algorithm.

Practice: Configuring and Observing Automatic In-Memory

This practice shows how to configure Automatic In-Memory and then observe how in-memory objects are automatically and dynamically populated in the IM column store without user intervention, and then possibly automatically evicted from the IM column store.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the `/home/oracle/labs/M104783GC10/AutoIM_setup.sh`. The shell script configures the IM column store to 110M,

creates NO INMEMORY tables in HR schema in PDB20, and finally inserts rows in HR tables.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/AutoIM_setup.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 10 10:38:54 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SHUTDOWN ABORT
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area  851440264 bytes
Fixed Size                  9573000 bytes
Variable Size             339738624 bytes
Database Buffers          377487360 bytes
Redo Buffers                7200768 bytes
In-Memory Area            117440512 bytes
Database mounted.
SQL> ALTER SYSTEM SET sga_target=812M SCOPE=spfile;

System altered.

SQL> ALTER SYSTEM SET inmemory_size=110M SCOPE=SPFILE;

System altered.

SQL> ALTER SYSTEM SET query_rewrite_integrity=stale_tolerated
SCOPE=SPFILE;

System altered.

SQL> SET ECHO OFF

System altered.

SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=LOW SCOPE=SPFILE;

System altered.

SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
```

```
ORACLE instance started.

Total System Global Area  851440264 bytes
Fixed Size                  9573000 bytes
Variable Size             339738624 bytes
Database Buffers          377487360 bytes
Redo Buffers                7200768 bytes
In-Memory Area            117440512 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN;

Pluggable database altered.

SQL> CONNECT sys/password@PDB20 AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=LOW SCOPE=SPFILE;

System altered.

SQL> ALTER SYSTEM SET query_rewrite_integrity=stale_tolerated
SCOPE=SPFILE;

System altered.

SQL> shutdown immediate
Pluggable Database closed.
SQL> STARTUP
Pluggable Database opened.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 10 10:41:05 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SYSTEM SET db_create_file_dest='/home/oracle/labs';

System altered.

SQL>
SQL> DROP TABLESPACE imtbs INCLUDING CONTENTS AND DATAFILES cascade
constraints;

Tablespace dropped.

SQL> CREATE TABLESPACE imtbs DATAFILE '/home/oracle/labs/imtbs1.dbf'
```

```
SIZE 10G;

Tablespace created.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 10 10:44:02 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0


specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:

PL/SQL procedure successfully completed.

User created.

User altered.

User altered.

Grant succeeded.

Grant succeeded.

Session altered.

Session altered.

Session altered.

******  Creating REGIONS table ....

Table created.

Index created.

Table altered.

******  Creating COUNTRIES table ....
```

```
Table created.

Table altered.

******  Creating LOCATIONS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating DEPARTMENTS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating JOBS table ....

Table created.

Index created.

Table altered.

******  Creating EMPLOYEES table ....

Table created.

Index created.

Table altered.

Table altered.

Sequence created.

******  Creating JOB_HISTORY table ....

Table created.

Index created.

Table altered.

******  Creating EMP_DETAILS_VIEW view ...
```

```
View created.

Commit complete.

Session altered.

******  Populating REGIONS table ....

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating COUNTIRES table ....

1 row created.
...
******  Populating LOCATIONS table ....

1 row created.
...

******  Populating DEPARTMENTS table ....

Table altered.

...
1 row created.

******  Populating JOBS table ....

1 row created.
...

******  Populating EMPLOYEES table ....

1 row created.
...

******  Populating JOB_HISTORY table ....

1 row created.
...

Table altered.

Commit complete.

Index created.

...
```

```
Commit complete.

Procedure created.

Trigger created.

Trigger altered.

Procedure created.

Trigger created.

Commit complete.

Comment created.

...

Commit complete.

PL/SQL procedure successfully completed.

SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 10 10:44:22 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE hr.emp CASCADE CONSTRAINTS;
DROP TABLE hr.emp CASCADE CONSTRAINTS
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> CREATE TABLE hr.emp INMEMORY AS SELECT * FROM hr.employees ;

Table created.

SQL> INSERT INTO hr.emp SELECT * FROM hr.emp;

107 rows created.

SQL> /

214 rows created.

SQL> /
```

```
428 rows created.

SQL> /

856 rows created.

SQL> /

1712 rows created.

SQL> /

3424 rows created.

SQL> /

6848 rows created.

SQL> /

13696 rows created.

SQL> /

27392 rows created.

SQL> /

54784 rows created.

SQL> /

109568 rows created.

SQL> /

219136 rows created.

SQL> /

438272 rows created.

SQL> /

876544 rows created.

SQL> /

1753088 rows created.

SQL> COMMIT;

Commit complete.
```

```
SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Query the data dictionary to determine whether HR tables are specified as INMEMORY.

```
$ sqlplus sys@PDB20 AS SYSDBA

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL> COL table_name FORMAT A18
SQL> SELECT table_name, inmemory, inmemory_compression
FROM   dba_tables WHERE  owner='HR';

TABLE_NAME         INMEMORY INMEMORY_COMPRESS
------------------ -------- -----------------
REGIONS            DISABLED
LOCATIONS          DISABLED
DEPARTMENTS        DISABLED
JOBS               DISABLED
EMPLOYEES          DISABLED
JOB_HISTORY        DISABLED
EMP                ENABLED  FOR QUERY LOW
COUNTRIES          DISABLED

8 rows selected.

SQL>
```

4. Apply the INMEMORY and MEMCOMPRESS FOR CAPACITY LOW attributes to the HR.JOB_HISTORY table.

```
SQL> ALTER TABLE hr.job_history INMEMORY MEMCOMPRESS FOR CAPACITY LOW;

Table altered.

SQL> SELECT table_name, inmemory, inmemory_compression
FROM   dba_tables WHERE  owner='HR';

TABLE_NAME         INMEMORY INMEMORY_COMPRESS
------------------ -------- -----------------
REGIONS            DISABLED
LOCATIONS          DISABLED
DEPARTMENTS        DISABLED
JOBS               DISABLED
EMPLOYEES          DISABLED
JOB_HISTORY        ENABLED  FOR CAPACITY LOW
EMP                ENABLED  FOR QUERY LOW
COUNTRIES          DISABLED
```

```
8 rows selected.

SQL>
```

5. Connect to the CDB root, then set `INMEMORY_AUTOMATIC_LEVEL` to `HIGH`, and re-start the database instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=HIGH SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  851442944 bytes
Fixed Size                  9571584 bytes
Variable Size             440401920 bytes
Database Buffers          276824064 bytes
Redo Buffers                7204864 bytes
In-Memory Area            117440512 bytes
Database mounted.
Database opened.
SQL>
```

6. Query the data dictionary to determine whether `HR` tables are specified as `INMEMORY`.

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password
Connected.
SQL> SELECT table_name, inmemory, inmemory_compression
FROM   dba_tables WHERE  owner='HR';

TABLE_NAME         INMEMORY INMEMORY_COMPRESS
------------------ -------- -----------------
REGIONS            DISABLED
LOCATIONS          DISABLED
DEPARTMENTS        DISABLED
JOBS               DISABLED
EMPLOYEES          DISABLED
JOB_HISTORY        ENABLED  FOR CAPACITY LOW
EMP                ENABLED  FOR QUERY LOW
COUNTRIES          DISABLED

8 rows selected.

SQL>
```

Why are the HR tables not enabled to INMEMORY, except those already manually set to INMEMORY? Display the INMEMORY_AUTOMATIC_LEVEL in the PDB.

```
SQL> SHOW PARAMETER INMEMORY_AUTOMATIC_LEVEL

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------
inmemory_automatic_level             string      LOW
SQL> SELECT ispdb_modifiable FROM v$parameter WHERE
name='inmemory_automatic_level';

ISPDB
-----
TRUE

SQL>
```

7. Set INMEMORY_AUTOMATIC_LEVEL to HIGH at the PDB level, and re-start PDB20.

```
SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=HIGH SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
Pluggable Database closed.
SQL> STARTUP
Pluggable Database opened.
SQL>
```

8. Wait one minute to observe the HR tables to be automatically assigned the INMEMORY attribute.

```
SQL> SELECT table_name, inmemory, inmemory_compression
FROM   dba_tables WHERE owner='HR';

TABLE_NAME         INMEMORY INMEMORY_COMPRESS
------------------ -------- -----------------
REGIONS            ENABLED  AUTO
LOCATIONS          ENABLED  AUTO
DEPARTMENTS        ENABLED  AUTO
JOBS               ENABLED  AUTO
EMPLOYEES          ENABLED  AUTO
JOB_HISTORY        ENABLED  FOR CAPACITY LOW
EMP                ENABLED  FOR QUERY LOW
COUNTRIES          DISABLED

8 rows selected.

SQL>
```

Observe that HR.JOB_HISTORY and HR.JOB_EMP which were manually specified as INMEMORY, retain their previous settings.

Why is HR.COUNTRIES not automatically enabled?

```
SQL> ALTER TABLE hr.countries INMEMORY;
 ALTER TABLE hr.countries INMEMORY
*
ERROR at line 1:
ORA-64358: in-memory column store feature not supported for IOTs


SQL>
```

9. Execute the /home/oracle/labs/M104783GC10/AutoIM_scan.sql SQL script to populate the HR tables into the IM Column Store.

```
SQL> @/home/oracle/labs/M104783GC10/AutoIM_scan.sql
SQL> SELECT /*+ FULL(hr.employees) NO_PARALLEL(hr.employees) */
count(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL> SELECT /*+ FULL(hr.departments) NO_PARALLEL(hr.departments) */
count(*) FROM hr.departments;

  COUNT(*)
----------
        27

SQL> SELECT /*+ FULL(hr.locations) NO_PARALLEL(hr.locations) */
count(*) FROM hr.locations;

  COUNT(*)
----------
        23

SQL> SELECT /*+ FULL(hr.jobs) NO_PARALLEL(hr.jobs) */ count(*) FROM
hr.jobs;

  COUNT(*)
----------
        19

SQL> SELECT /*+ FULL(hr.regions) NO_PARALLEL(hr.regions) */ count(*)
FROM hr.regions;

  COUNT(*)
----------
         4

SQL> SELECT /*+ FULL(hr.emp) NO_PARALLEL(hr.emp) */ count(*) FROM
hr.emp;

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

```
   3506176

SQL>
```

10. Display the population status of the HR tables into the IM Column Store.

```
SQL> COL segment_name FORMAT A12
SQL> SELECT segment_name, inmemory_size, bytes_not_populated,
inmemory_compression FROM v$im_segments;

SEGMENT_NAME INMEMORY_SIZE BYTES_NOT_POPULATED INMEMORY_COMPRESS
------------ ------------- ------------------- -----------------
EMP               44433408                   0 FOR QUERY LOW

SQL>
```

Why aren't the ENABLED AUTO tables not populated into the IM column store? The internal statistics are not sufficient yet to identify cold and hot data in the IM column store to consider which segments can be populated into the IM column store.

11. Execute the /home/oracle/labs/M104783GC10/AutoIM_scan_AUTO.sql SQL script to insert more rows into HR.EMPLOYEES table, query the HR.EMPLOYEES table and possibly then get the table automatically populated into the IM column store.

```
SQL> @/home/oracle/labs/M104783GC10/AutoIM_scan_AUTO.sql
SQL> set echo on
SQL> begin
  2  for i in (select constraint_name, table_name from dba_constraints
where table_name='EMPLOYEES') LOOP
  3  execute immediate 'alter table hr.employees drop constraint '||
i.constraint_name||' CASCADE';
  4  end loop;
  5  end;
  6  /

PL/SQL procedure successfully completed.

SQL> drop index hr.EMP_EMP_ID_PK;
drop index hr.EMP_EMP_ID_PK
             *
ERROR at line 1:
ORA-01418: specified index does not exist


SQL>
SQL> INSERT INTO hr.employees SELECT * FROM hr.employees;

107 rows created.

SQL> /

214 rows created.

SQL> /
```

```
428 rows created.

SQL> /

856 rows created.

SQL> /

1712 rows created.

SQL> /

3424 rows created.

SQL> /

6848 rows created.

SQL> /

13696 rows created.

SQL> /

27392 rows created.

SQL> COMMIT;

Commit complete.

SQL> /

Commit complete.

SQL> /

Commit complete.

SQL> /

Commit complete.

SQL> /

Commit complete.

SQL> /

Commit complete.

SQL> /

Commit complete.
```

```
SQL> COMMIT;

Commit complete.

SQL>
```

12. Display the population status of the HR tables into the IM Column Store. You may have to wait for a few minutes before the population of EMPLOYEES table starts.

```
SQL> SELECT segment_name, inmemory_size, bytes_not_populated,
inmemory_compression FROM v$im_segments;

SEGMENT_NAME INMEMORY_SIZE BYTES_NOT_POPULATED INMEMORY_COMPRESS
------------ ------------- ------------------- -----------------
EMP               44433408                   0 FOR QUERY LOW
EMPLOYEES          1310720                   0 AUTO

SQL> EXIT
$
```

Observe the HR.EMPLOYEES table is now populated with an INMEMORY_COMPRESS value set to AUTO. Compression used the automatic in-memory management based on internal statistics. After some time, the HR.EMP may be evicted according to the internal statistics. If you re-query the HR.EMP table, the statistics may decide to evict the HR.EMPLOYEES to let the HR.EMP populate back into the IM column store.

In-Memory Hybrid Scans

Oracle Database supports In-memory scans when not all columns in a table have been populated into the In-Memory Column Store (IM column store).

This situation can occur when columns have been specified as NO INMEMORY to save space. In-memory hybrid scans can access some data from the IM column store, and some data from the row store, improving performance by orders of magnitude over pure row store queries.

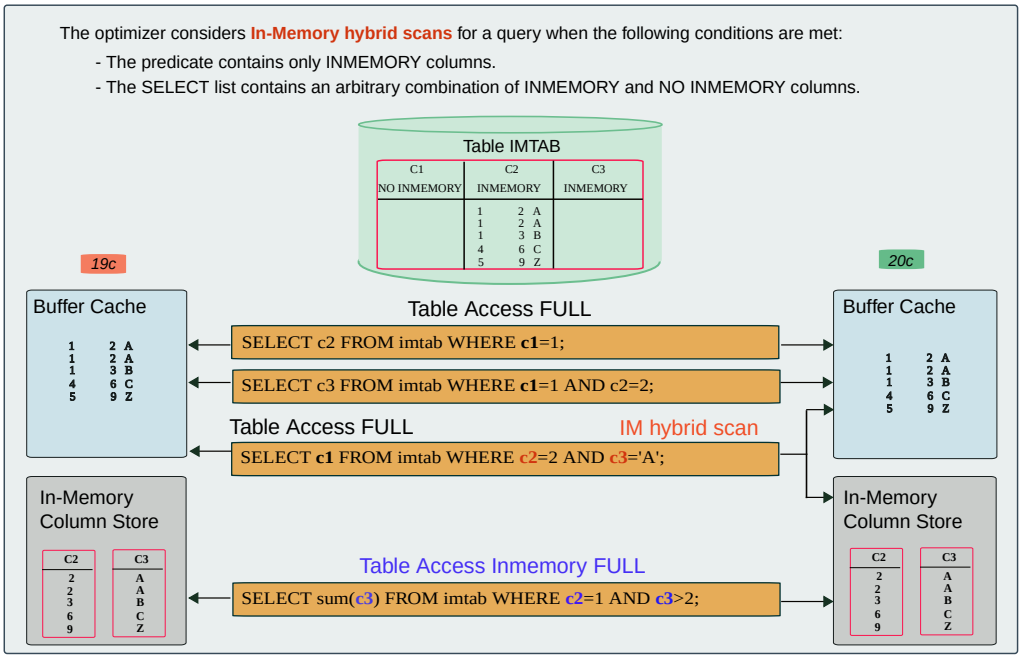
- Details: In-Memory Hybrid Scans
  This page provides more detailed information about queries referencing both INMEMORY and NO INMEMORY columns behaving differently in Oracle Database 20c.
- Practice: Using In-Memory Hybrid Scans in Queries
  This practice shows how queries referencing both INMEMORY and NO INMEMORY columns can access columnar data. This optimizer access method called IM hybrid scan can improve performance by orders of magnitude. If the optimizer chooses a table scan, the storage engine automatically determines whether an IM hybrid scan performs better than a regular row store scan from the buffer cache.

Related Topics

- *Oracle® Database In-Memory Guide*

Details: In-Memory Hybrid Scans

This page provides more detailed information about queries referencing both INMEMORY and NO INMEMORY columns behaving differently in Oracle Database 20c.



Before Oracle Database 20c, if a query referenced any column with the `NO INMEMORY` attribute, then the query accessed all data from the row store (buffer cache). Therefore, the table scan could not take advantage of columnar formats, predicate pushdown, and other In-Memory features.

Starting in Oracle Database 20c, queries that reference both `INMEMORY` and `NO INMEMORY` columns can access columnar data.

In some cases, an IM hybrid scan can improve performance by orders of magnitude. The greatest performance benefits occur when a query has selective filters. In this case, the IM column store can quickly filter out most rows so that the row store projects only a small number of rows.

To achieve optimal performance, the optimizer compares different access methods. If the optimizer chooses a table scan, then the storage engine automatically determines whether an IM hybrid scan performs better than a regular row store scan. The optimizer considers hybrid scans when the following conditions are met:

- The predicate contains only `INMEMORY` columns.
- The `SELECT` list contains an arbitrary combination of `INMEMORY` and `NO INMEMORY` columns.

An IM hybrid scan logically divides the work into two: one part processes the query on the IM column store, and the other part processes the query on the row store. In the execution plan, the operation named `TABLE ACCESS INMEMORY FULL (HYBRID)` indicates a hybrid scan. Note that if runtime statistics indicate that performance will be faster by accessing the row store only, then the database can disable the IM hybrid scan at runtime.

Practice: Using In-Memory Hybrid Scans in Queries

This practice shows how queries referencing both `INMEMORY` and `NO INMEMORY` columns can access columnar data. This optimizer access method called IM hybrid scan can improve performance by orders of magnitude. If the optimizer chooses a table scan,

the storage engine automatically determines whether an IM hybrid scan performs better than a regular row store scan from the buffer cache.

1. Before starting any new practice, refer to the practices environment recommendations.
2. The optimizer considers hybrid scans when the following conditions are met:
   - The predicate contains only `INMEMORY` columns.
   - The SELECT list contains an arbitrary combination of `INMEMORY` and `NO INMEMORY` columns.
3. Before testing the queries on in-memory tables containing `INMEMORY` and `NO INMEMORY` columns, execute the `/home/oracle/labs/M104783GC10/IM_Hybrid_setup.sh`. The shell script configures the IM column store to 110M, creates an in-memory table `IMU.IMTAB` containing two `INMEMORY` columns and one `NO INMEMORY` column, and finally inserts rows in the table. The shell script executes the same operations in an Oracle Database 19c and Oracle Database 20c.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/IM_Hybrid_setup.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Thu Jan 9 03:51:59 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SHUTDOWN ABORT
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area  851442944 bytes
Fixed Size                  9571584 bytes
Variable Size             331350016 bytes
Database Buffers          385875968 bytes
Redo Buffers                7204864 bytes
In-Memory Area            117440512 bytes
Database mounted.
SQL> ALTER SYSTEM SET sga_target=812M SCOPE=spfile;

System altered.

SQL> ALTER SYSTEM SET inmemory_size=110M SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
ORA-01109: database not open


Database dismounted.
```

```
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  851442944 bytes
Fixed Size                  9571584 bytes
Variable Size             331350016 bytes
Database Buffers          385875968 bytes
Redo Buffers                7204864 bytes
In-Memory Area            117440512 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Thu Jan 9 03:53:36 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SYSTEM SET db_create_file_dest='';

System altered.

SQL> DROP USER imu CASCADE;

User dropped.

SQL> DROP TABLESPACE imtbs INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

SQL> CREATE TABLESPACE imtbs DATAFILE '/home/oracle/labs/imtbs1.dbf'
SIZE 500M;

Tablespace created.

SQL> CREATE USER imu IDENTIFIED BY password DEFAULT TABLESPACE imtbs;

User created.

SQL> GRANT create session, create table, unlimited tablespace TO imu;

Grant succeeded.
```

```
SQL>
SQL> CREATE TABLE imu.imtab (c1_noinmem NUMBER, c2_inmem NUMBER,
c3_inmem VARCHAR2(4000))
  2         INMEMORY PRIORITY high MEMCOMPRESS for capacity low NO
INMEMORY(c1_noinmem);

Table created.

SQL> INSERT INTO imu.imtab VALUES (3,4,'Test20c');

1 row created.

SQL> INSERT INTO imu.imtab SELECT c1_noinmem + (select max(c1_noinmem)
from imu.imtab),
  2                               c2_inmem + (select max(c2_inmem) from
imu.imtab),
  3                               c3_inmem|| (select max(c2_inmem) from
imu.imtab) FROM imu.imtab;

1 row created.

SQL> /

2 rows created.

SQL> /

4 rows created.

SQL> /

8 rows created.

SQL> /

16 rows created.

SQL> /

32 rows created.

SQL> /

64 rows created.

SQL> /

128 rows created.

SQL> /

256 rows created.

SQL> /
```

```
512 rows created.

SQL> /

1024 rows created.

SQL> /

2048 rows created.

SQL> /

4096 rows created.

SQL> /

8192 rows created.

SQL> /

16384 rows created.

SQL> /

32768 rows created.

SQL> /

65536 rows created.

SQL> /

131072 rows created.

SQL> COMMIT;

Commit complete.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

4. Connect to PDB20 as SYSTEM and set formats for the queried columns.

```
$ sqlplus system@PDB20
SQL*Plus: Release 20.0.0.0.0 - Development on Thu Jan 9 04:08:41 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Wed Jan 08 2020 12:03:56 +00:00
```

```
Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL> COL table_name FORMAT A10
SQL> COL inmemory_compression FORMAT A11
SQL> COL COL_NO_INMEM FORMAT 99999999999999999999
SQL> COL COL_INMEM FORMAT 99999999999999999999
SQL> COL segment_name FORMAT A12
SQL>
```

5. Display the in-memory attributes of the `IMU.IMTAB` table and of all columns of the table.

```
SQL> SELECT table_name, inmemory_compression "COMPRESSION",
inmemory_priority "PRIORITY"
FROM   dba_tables WHERE owner='IMU';

TABLE_NAME COMPRESSION       PRIORITY
---------- ----------------- --------
IMTAB      FOR CAPACITY LOW  HIGH

SQL> SELECT obj_num, segment_column_id, inmemory_compression FROM
v$im_column_level im, dba_objects o
WHERE  im.obj_num = o.object_id
AND    o.object_name='IMTAB';

   OBJ_NUM SEGMENT_COLUMN_ID INMEMORY_CO
---------- ---------------- -----------
     74869                1 NO INMEMORY
     74869                2 DEFAULT
     74869                3 DEFAULT

SQL>
```

6. Execute a full scan on the `IMU.IMTAB` table so as to populate the table into the IM Column Store.

```
SQL> SELECT /*+ FULL(imu.imtab) NO_PARALLEL(imu.imtab) */ COUNT(*) FROM
imu.imtab;

  COUNT(*)
----------
    262144

SQL>
```

7. Verify that the `IMU.IMTAB` table is populated into the IM Column Store.

```
SQL> COL segment_name FORMAT A12
SQL> SELECT segment_name, bytes, inmemory_size, bytes_not_populated
FROM   v$im_segments;
```

```
SEGMENT_NAME      BYTES INMEMORY_SIZE BYTES_NOT_POPULATED
------------ ---------- ------------ ------------------
IMTAB          17481728       4456448                  0

SQL>
```

8. Execute a first query on the `IMU.IMTAB` table. The SELECT list contains the `NO INMEMORY` column and the predicate contains only the `NO INMEMORY` columns. Then examine the execution plan.

```
SQL> SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab
WHERE  c1_noinmem BETWEEN 5 AND 1258291;

           COL_NO_INMEM
-----------------------
           103079608317

SQL> SELECT * FROM table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
SQL_ID  1dpya5ws8gbvx, child number 0
-------------------------------------
SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab WHERE  c1_noin
mem BETWEEN 5 AND 1258291

Plan hash value: 360700294
------------------------------------------------------------------------
----
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)|
Time     |
------------------------------------------------------------------------
----
|   0 | SELECT STATEMENT   |       |       |       |   547 (100)|
|
|   1 |  SORT AGGREGATE    |       |     1 |    13 |            |
|
|*  2 |   TABLE ACCESS FULL| IMTAB |   292K|  3712K|   547   (1)| 00 :
00:01 |
------------------------------------------------------------------------
----
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("C1_NOINMEM">=5 AND "C1_NOINMEM"<=1258291))
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

24 rows selected.

SQL>
```

The optimizer in both sessions choose the `TABLE ACCESS FULL` method because the predicate does not contain only `INMEMORY` columns.

9. Execute a second query on the `IMU.IMTAB` table. The SELECT list contains the `NO INMEMORY` column and the predicate contains both a `NO INMEMORY` column and an `INMEMORY` column. Then examine the execution plan.

```
SQL> SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab
WHERE  c1_noinmem BETWEEN 5 AND 1258291 AND c3_inmem LIKE 'Test20c%';

           COL_NO_INMEM
----------------------
           103079608317

SQL> SELECT * FROM table(dbms_xplan.display_cursor());
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
SQL_ID  afz9bm3rscr3y, child number 0
-------------------------------------
SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab WHERE  c1_noinmem
BETWEEN 5 AND 1258291 AND c3_inmem LIKE 'Test20c%'

Plan hash value: 360700294
------------------------------------------------------------------------
----
| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)|
Time     |
------------------------------------------------------------------------
----
|   0 | SELECT STATEMENT   |       |       |       |   582 (100)|
|
|   1 |  SORT AGGREGATE    |       |     1 |  2015 |            |
|
|*  2 |   TABLE ACCESS FULL| IMTAB |   230K|   443M|   582   (1)|
00:00:01 |
------------------------------------------------------------------------
----

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(("C1_NOINMEM">=5 AND "C1_NOINMEM"<=1258291 AND "C3_INMEM"
LIKE 'Test20c%'))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


25 rows selected.

SQL>
```

The optimizer in both sessions choose the `TABLE ACCESS FULL` access method because the predicate does not contain only `INMEMORY` columns. It contains a `INMEMORY` column and an `NO INMEMORY` columns.

10. Execute a third query on the `IMU.IMTAB` table. The SELECT list contains the `NO INMEMORY` column and the predicate contains only `INMEMORY` columns. Then examine the execution plan.

```
SQL> SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab
WHERE  c2_inmem BETWEEN 5 AND 1258291 AND c3_inmem LIKE 'Test20c%';

           COL_NO_INMEM
----------------------
           103079608317

SQL> SELECT * FROM table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
SQL_ID  f07n4gc330rhz, child number 0
-------------------------------------
SELECT sum(c1_noinmem) AS COL_NO_INMEM FROM imu.imtab WHERE  c2_inmem
BETWEEN 5 AND 1258291 AND c3_inmem LIKE 'Test20c%'

Plan hash value: 360700294

-------------------------------------------------------------------------
-----------
| Id  | Operation                           | Name  | Rows  | Bytes |
Cost (%CPU)| Time     |
-------------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT                    |       |       | |   582
(100)|          |
|   1 |  SORT AGGREGATE                     |       |     1 |  2028
|            |          |
|*  2 |   TABLE ACCESS INMEMORY FULL (HYBRID)| IMTAB |   230K|
445M|   582   (1)| 00:00:01 |
-------------------------------------------------------------------------
-----

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("C2_INMEM">=5 AND "C2_INMEM"<=1258291 AND "C3_INMEM"
LIKE 'Test20c%'))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

24 rows selected.

SQL>
```

The optimizer in both sessions choose different access methods. In 20c, the `TABLE ACCESS INMEMORY FULL (HYBRID)` access method is chosen because the predicate contains only `INMEMORY` columns and the SELECT list a `NO INMEMORY` column.

11. Drop the IMU user.

```
SQL> DROP USER imu CASCADE;

User dropped.

SQL> EXIT
$
```

Database In-Memory External Table Enhancements

For a partitioned or hybrid external table, the INMEMORY clause is supported at both the table and partition level. For hybrid tables, the table-level INMEMORY attribute applies to all partitions, whether internal or external.

This enhancement significantly broadens support for in-memory external tables.

- Practice: Using In-Memory With Hybrid Partitioned Tables
  This practice shows how the INMEMORY attribute on a hybrid partitioned table is handled at both the table and partition level, whether internal or external partitions.

Related Topics

- *Oracle® Database In-Memory Guide*

Practice: Using In-Memory With Hybrid Partitioned Tables

This practice shows how the INMEMORY attribute on a hybrid partitioned table is handled at both the table and partition level, whether internal or external partitions.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the /home/oracle/labs/M104784GC10/IM_Hybrid_External_setup.sh shell script. The shell script configures the IM column store to 110M, creates the HYPTEXT user and directories for external files.

```
$ cd /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/IM_Hybrid_External_setup.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Jan 15 05:17:45 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SHUTDOWN ABORT
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area  851442944 bytes
Fixed Size                  9571584 bytes
Variable Size             432013312 bytes
```

```
Database Buffers          285212672 bytes
Redo Buffers                7204864 bytes
In-Memory Area            117440512 bytes
Database mounted.
SQL> ALTER SYSTEM SET sga_target=812M SCOPE=spfile;

System altered.

SQL> ALTER SYSTEM SET inmemory_size=110M SCOPE=SPFILE;

System altered.

SQL> ALTER SYSTEM SET query_rewrite_integrity=stale_tolerated
SCOPE=SPFILE;

System altered.

SQL> SET ECHO OFF

System altered.

SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=LOW SCOPE=SPFILE;

System altered.

SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  851442944 bytes
Fixed Size                  9571584 bytes
Variable Size             432013312 bytes
Database Buffers          285212672 bytes
Redo Buffers                7204864 bytes
In-Memory Area            117440512 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN;

Pluggable database altered.

SQL> CONNECT sys/password@PDB20 AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=LOW SCOPE=SPFILE;

System altered.

SQL> ALTER SYSTEM SET query_rewrite_integrity=stale_tolerated
SCOPE=SPFILE;
```

```
System altered.

SQL> shutdown immediate
Pluggable Database closed.
SQL> STARTUP
Pluggable Database opened.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Jan 15 05:19:12 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SYSTEM SET db_create_file_dest='';

System altered.

SQL> DROP USER hyptext CASCADE;
DROP USER hyptext CASCADE
          *
ERROR at line 1:
ORA-01918: user 'HYPTEXT' does not exist


SQL> DROP TABLESPACE imtbs INCLUDING CONTENTS AND DATAFILES cascade
constraints;
DROP TABLESPACE imtbs INCLUDING CONTENTS AND DATAFILES cascade
constraints
*
ERROR at line 1:
ORA-00959: tablespace 'IMTBS' does not exist


SQL> CREATE TABLESPACE imtbs DATAFILE '/u02/app/oracle/oradata/
imtbs1.dbf' SIZE 10G;

Tablespace created.

SQL> CREATE USER hyptext IDENTIFIED BY password DEFAULT TABLESPACE
imtbs;

User created.

SQL> GRANT create session, create table, unlimited tablespace TO
hyptext;

Grant succeeded.
```

```
SQL> HOST mkdir -p /home/oracle/labs/M104784GC10/CENT17

SQL> HOST mkdir -p /home/oracle/labs/M104784GC10/CENT18

SQL> HOST mkdir -p /home/oracle/labs/M104784GC10/CENT19

SQL> HOST mkdir -p /home/oracle/labs/M104784GC10/CENT20

SQL> HOST mv /home/oracle/labs/M104784GC10/cent17.dat /home/oracle/labs/
M104784GC10/CENT17

SQL> HOST mv /home/oracle/labs/M104784GC10/cent19.dat /home/oracle/labs/
M104784GC10/CENT19

SQL> HOST mv /home/oracle/labs/M104784GC10/cent20.dat /home/oracle/labs/
M104784GC10/CENT20

SQL> CREATE OR REPLACE DIRECTORY cent17 AS '/home/oracle/labs/
M104784GC10/CENT17';

Directory created.

SQL> CREATE OR REPLACE DIRECTORY cent18 AS '/home/oracle/labs/
M104784GC10/CENT18';

Directory created.

SQL> CREATE OR REPLACE DIRECTORY cent19 AS '/home/oracle/labs/
M104784GC10/CENT19';

Directory created.

SQL> CREATE OR REPLACE DIRECTORY cent20 AS '/home/oracle/labs/
M104784GC10/CENT20';

Directory created.

SQL> GRANT read, write ON DIRECTORY cent17 TO hyptext;

Grant succeeded.

SQL> GRANT read, write ON DIRECTORY cent18 TO hyptext;

Grant succeeded.

SQL> GRANT read, write ON DIRECTORY cent19 TO hyptext;

Grant succeeded.

SQL> GRANT read, write ON DIRECTORY cent20 TO hyptext;

Grant succeeded.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
```

```
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. The first part of the practice shows how the INMEMORY attribute on a hybrid partitioned table is handled at the table level with internal or external partitions.

   a. Connect to PDB20 as SYSTEM.

```
$ sqlplus system@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Thu Jan 9 04:08:41 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Wed Jan 08 2020 12:03:56 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0
SQL> COL partition_name FORMAT A14
SQL> COL segment_name FORMAT A14
SQL>
```

   b. Create an in-memory hybrid partitioned table. Apply the INMEMORY attribute at the table level.

```
SQL> CREATE TABLE hyptext.inmem_tab
  (history_event NUMBER , time_id DATE) TABLESPACE imtbs
   EXTERNAL PARTITION ATTRIBUTES
   (TYPE ORACLE_LOADER DEFAULT DIRECTORY cent20
        ACCESS PARAMETERS
         (FIELDS TERMINATED BY ',' (history_event , time_id DATE
'dd-MON-yyyy'))
         REJECT LIMIT UNLIMITED
   )
  PARTITION BY RANGE (time_id)
  (PARTITION cent18 VALUES LESS THAN (TO_DATE('01-Jan-1800','dd-MON-
yyyy')) EXTERNAL,
   PARTITION cent19 VALUES LESS THAN (TO_DATE('01-Jan-1900','dd-MON-
yyyy')) EXTERNAL
                    DEFAULT DIRECTORY cent19 LOCATION
('cent19.dat'),
   PARTITION cent20 VALUES LESS THAN (TO_DATE('01-Jan-2000','dd-MON-
yyyy')) EXTERNAL
                    LOCATION('cent20.dat'),
   PARTITION y2000 VALUES LESS THAN (TO_DATE('01-Jan-2001','dd-MON-
yyyy')),
   PARTITION pmax VALUES LESS THAN (MAXVALUE))
  INMEMORY MEMCOMPRESS FOR QUERY HIGH;

Table created.
```

```
SQL>
```

c. Which partitions are defined as in-memory segments?

```
SQL> SELECT partition_name, inmemory, inmemory_compression
FROM   dba_tab_partitions
WHERE  table_name = 'INMEM_TAB';

PARTITION_NAME INMEMORY INMEMORY_COMPRESS
-------------- -------- -----------------
CENT18
CENT19
CENT20
PMAX           ENABLED  FOR QUERY HIGH
Y2000          ENABLED  FOR QUERY HIGH

SQL>
```

Internal partitions are defined as in-memory. External partitions are not defined as in-memory, nor as no in-memory.

Use the `DBA_XTERNAL_TAB_PARTITIONS` view to show in-memory status on external partitions.

```
SQL> SELECT partition_name, inmemory, inmemory_compression
             FROM dba_xternal_tab_partitions WHERE
TABLE_NAME='INMEM_TAB';

PARTITION_NAME INMEMORY INMEMORY_COMPRESS
-------------- -------- -----------------
CENT19         ENABLED  FOR QUERY HIGH
CENT20         ENABLED  FOR QUERY HIGH

SQL>
```

d. Execute the `/home/oracle/labs/M104784GC10/insert_select.sql` SQL script. The script inserts rows into the partitions of the table and query the table to populate the data into the in-memory column store. Which partitions are populated into the in-memory column store?

```
SQL> @/home/oracle/labs/M104784GC10/insert_select.sql
SQL> INSERT INTO hyptext.inmem_tab VALUES (21,to_date('31.12.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (22,to_date('31.10.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (23,to_date('01.02.2000',
'dd.mm.yyyy'));
```

```
1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (24,to_date('27.03.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (25,to_date('31.03.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (26,to_date('15.04.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (27,to_date('02.09.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (29,to_date('12.08.2018',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (30,to_date('15.09.2017',
'dd.mm.yyyy'));

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM hyptext.inmem_tab;

HISTORY_EVENT TIME_ID
------------- ---------
           11 01-JAN-76
           12 01-JAN-15
           13 01-JAN-28
           14 01-JAN-37
           15 01-JAN-49
           16 01-FEB-59
           17 01-FEB-96
           18 01-FEB-97
           19 01-FEB-98
           20 01-FEB-98
            1 01-JAN-76
            2 01-JAN-15
            3 01-JAN-28
            4 01-JAN-37
```

```
         5 01-JAN-49
         6 01-FEB-59
         7 01-FEB-96
         8 01-FEB-97
         9 01-FEB-98
        10 01-FEB-98
        21 31-DEC-00
        22 31-OCT-00
        23 01-FEB-00
        24 27-MAR-00
        25 31-MAR-00
        26 15-APR-00
        27 02-SEP-00
        29 12-AUG-18
        30 15-SEP-17

29 rows selected.

SQL>


SQL> SELECT segment_name, partition_name FROM v$im_segments;

SEGMENT_NAME   PARTITION_NAME
-------------- --------------
INMEM_TAB      PMAX
INMEM_TAB      CENT19
INMEM_TAB      Y2000
INMEM_TAB      CENT20

SQL19>
```

All internal and external partitions are populated into the in-memory column store because the `INMEMORY` attribute was set at the table level.

e. Does the execution plan show the different types of access to partitions?

```
SQL> EXPLAIN PLAN FOR SELECT * FROM hyptext.inmem_tab;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 2513257138
------------------------------------------------------------------------
| Id  | Operation                                | Name      | Rows
| Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |           |   188K
|  4057K|   105   (1)| 00:00:01 |       |       |
|   1 |  PARTITION RANGE ALL                     |           |   188K
|  4057K|   105   (1)| 00:00:01 |     1 |     5 |
|   2 |   TABLE ACCESS HYBRID PART INMEMORY FULL| INMEM_TAB |   188K
```

```
|  4057K|   105   (1)| 00:00:01 |     1 |     5 |
|   3 |    TABLE ACCESS INMEMORY FULL           | INMEM_TAB |
|       |            |          |     1 |     5 |
--------------------------------------------------------------------

10 rows selected.

SQL> EXPLAIN PLAN FOR SELECT * FROM hyptext.inmem_tab PARTITION
(CENT19);

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
Plan hash value: 938963152
--------------------------------------------------------------------
| Id  | Operation                           | Name      | Rows  | B
ytes | Cost (%CPU)| Time     | Pstart| Pstop |
--------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |           |  8169 |
 175K|    29   (0)| 00:00:01 |       |       |
|   1 |  PARTITION RANGE SINGLE             |           |  8169 |
 175K|    29   (0)| 00:00:01 |     2 |     2 |
|   2 |   EXTERNAL TABLE ACCESS INMEMORY FULL| INMEM_TAB |  8169 |
 175K|    29   (0)| 00:00:01 |     2 |     2 |
--------------------------------------------------------------------

9 rows selected.

SQL>
```

The access path shows either `TABLE ACCESS HYBRID PART INMEMORY FULL` (`NO INMEMORY` and `INMEMORY` accessed segments) or `EXTERNAL TABLE ACCESS INMEMORY FULL` (`INMEMORY`) on the selected external partition.

f. Drop the `HYPTEXT.INMEM_TAB` table.

```
SQL> DROP TABLE hyptext.inmem_tab;

Table dropped.

SQL>
```

4. The second part of the practice shows how the `INMEMORY` attribute on internal or external partitions of a hybrid partitioned table is handled.

   a. Create an in-memory hybrid partitioned table. Apply the `INMEMORY` attribute at the partition level, on an internal and an external partition.

```
SQL> CREATE TABLE hyptext.inmem_tab
  (history_event NUMBER , time_id DATE) TABLESPACE imtbs
   EXTERNAL PARTITION ATTRIBUTES
   (TYPE ORACLE_LOADER DEFAULT DIRECTORY cent20
```

```
        ACCESS PARAMETERS
         (FIELDS TERMINATED BY ',' (history_event , time_id DATE
'dd-MON-yyyy'))
         REJECT LIMIT UNLIMITED
   )
  PARTITION BY RANGE (time_id)
  (PARTITION cent18 VALUES LESS THAN (TO_DATE('01-Jan-1800','dd-MON-
yyyy')) EXTERNAL,
   PARTITION cent19 VALUES LESS THAN (TO_DATE('01-Jan-1900','dd-MON-
yyyy')) EXTERNAL
                     DEFAULT DIRECTORY cent19 LOCATION
('cent19.dat')
                        INMEMORY MEMCOMPRESS FOR QUERY HIGH,
   PARTITION cent20 VALUES LESS THAN (TO_DATE('01-Jan-2000','dd-MON-
yyyy')) EXTERNAL
                     LOCATION('cent20.dat'),
   PARTITION y2000 VALUES LESS THAN (TO_DATE('01-Jan-2001','dd-MON-
yyyy'))
                        INMEMORY MEMCOMPRESS FOR CAPACITY LOW,
   PARTITION pmax VALUES LESS THAN (MAXVALUE));

Table created.

SQL>
```

b. Which partitions are defined as in-memory?

```
SQL> SELECT partition_name, inmemory, inmemory_compression
FROM   dba_tab_partitions
WHERE  table_name = 'INMEM_TAB';

PARTITION_NAME INMEMORY INMEMORY_COMPRESS
-------------- -------- -----------------
CENT18
CENT19
CENT20
PMAX           DISABLED
Y2000          ENABLED  FOR CAPACITY LOW

SQL>
```

Only internal partitions for which the `INMEMORY` attribute was set are defined as in-memory. External partitions, even those for which the `INMEMORY` attribute was set, are not defined as in-memory, nor as no in-memory.

Use the `DBA_XTERNAL_TAB_PARTITIONS` view to show in-memory status on external partitions.

```
SQL> SELECT partition_name, inmemory, inmemory_compression
              FROM dba_xternal_tab_partitions WHERE
TABLE_NAME='INMEM_TAB';

PARTITION_NAME INMEMORY INMEMORY_COMPRESS
-------------- -------- -----------------
CENT19         ENABLED  FOR QUERY HIGH
```

```
CENT20          DISABLED

SQL>
```

c. Execute the `/home/oracle/labs/M104784GC10/insert_select.sql` SQL script. The script inserts rows into the partitions of the table and query the table to populate the data into the in-memory column store. Which partitions are populated into the in-memory column store?

```
SQL> @/home/oracle/labs/M104784GC10/insert_select.sql
SQL> SET ECHO ON
SQL>
SQL> INSERT INTO hyptext.inmem_tab VALUES (21,to_date('31.12.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (22,to_date('31.10.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (23,to_date('01.02.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (24,to_date('27.03.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (25,to_date('31.03.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (26,to_date('15.04.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (27,to_date('02.09.2000',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (29,to_date('12.08.2018',
'dd.mm.yyyy'));

1 row created.

SQL> INSERT INTO hyptext.inmem_tab VALUES (30,to_date('15.09.2017',
'dd.mm.yyyy'));
```

```
1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM hyptext.inmem_tab;

HISTORY_EVENT TIME_ID
------------- ---------
           11 01-JAN-76
           12 01-JAN-15
           13 01-JAN-28
           14 01-JAN-37
           15 01-JAN-49
           16 01-FEB-59
           17 01-FEB-96
           18 01-FEB-97
           19 01-FEB-98
           20 01-FEB-98
            1 01-JAN-76
            2 01-JAN-15
            3 01-JAN-28
            4 01-JAN-37
            5 01-JAN-49
            6 01-FEB-59
            7 01-FEB-96
            8 01-FEB-97
            9 01-FEB-98
           10 01-FEB-98
           21 31-DEC-00
           22 31-OCT-00
           23 01-FEB-00
           24 27-MAR-00
           25 31-MAR-00
           26 15-APR-00
           27 02-SEP-00
           29 12-AUG-18
           30 15-SEP-17

29 rows selected.

SQL>


SQL> SELECT segment_name, partition_name FROM v$im_segments;

SEGMENT_NAME   PARTITION_NAME
-------------- --------------
INMEM_TAB      CENT19
INMEM_TAB      Y2000

SQL>
```

In the Oracle Database 20c session, internal and external partitions defined as in-memory are populated into the in-memory column store.

d. Does the execution plan show the different types of access to partitions?

```
SQL> EXPLAIN PLAN FOR SELECT * FROM hyptext.inmem_tab;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 2513257138
------------------------------------------------------------------------
| Id  | Operation                                | Name      | Rows
| Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |           |   188K
|  4057K|   367   (1)| 00:00:01 |       |       |
|   1 |  PARTITION RANGE ALL                     |           |   188K
|  4057K|   367   (1)| 00:00:01 |     1 |     5 |
|   2 |   TABLE ACCESS HYBRID PART INMEMORY FULL| INMEM_TAB |   188K
|  4057K|   367   (1)| 00:00:01 |     1 |     5 |
|   3 |    TABLE ACCESS INMEMORY FULL            | INMEM_TAB |
|       |            |          |     1 |     5 |
------------------------------------------------------------------------

10 rows selected.

SQL> EXPLAIN PLAN FOR SELECT * FROM hyptext.inmem_tab PARTITION
(CENT19);

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 938963152
------------------------------------------------------------------------
| Id  | Operation                              | Name      | Rows  | B
ytes | Cost (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |           |  8169 |
 175K|    29   (0)| 00:00:01 |       |       |
|   1 |  PARTITION RANGE SINGLE                |           |  8169 |
 175K|    29   (0)| 00:00:01 |     2 |     2 |
|   2 |   EXTERNAL TABLE ACCESS INMEMORY FULL| INMEM_TAB |  8169 |
 175K|    29   (0)| 00:00:01 |     2 |     2 |
------------------------------------------------------------------------

9 rows selected.
```

```
SQL>


SQL> EXPLAIN PLAN FOR SELECT * FROM hyptext.inmem_tab PARTITION
(CENT20);

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 938963152
------------------------------------------------------------------------
| Id  | Operation                   | Name      | Rows  | Bytes | Co
st (%CPU)| Time     | Pstart| Pstop |
------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |           |  8169 |   175K|
 29   (0)| 00:00:01 |       |       |
|   1 |  PARTITION RANGE SINGLE     |           |  8169 |   175K|
 29   (0)| 00:00:01 |     3 |     3 |
|   2 |   EXTERNAL TABLE ACCESS FULL| INMEM_TAB |  8169 |   175K|
 29   (0)| 00:00:01 |     3 |     3 |
------------------------------------------------------------------------

9 rows selected.

SQL>
```

The access path shows either `TABLE ACCESS HYBRID PART INMEMORY FULL` (`NO INMEMORY` and `INMEMORY` accessed segments) or `EXTERNAL TABLE ACCESS INMEMORY FULL` (`INMEMORY`) on the selected external partition or `EXTERNAL TABLE ACCESS FULL` (`NO INMEMORY`) on the selected external partition.

e. Drop the `HYPTEXT.INMEM_TAB` table.

```
SQL> DROP TABLE HYPTEXT.INMEM_TAB PURGE;

Table dropped.

SQL> EXIT
$
```

Flashback

- PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past

PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past

PDBs can be recovered to an orphan PDB incarnation within the same CDB incarnation or an ancestor incarnation.

Availability of PDBs is enhanced. Both flashback and point-in-time recovery operations are supported when recovering PDBs to orphan PDB incarnations.

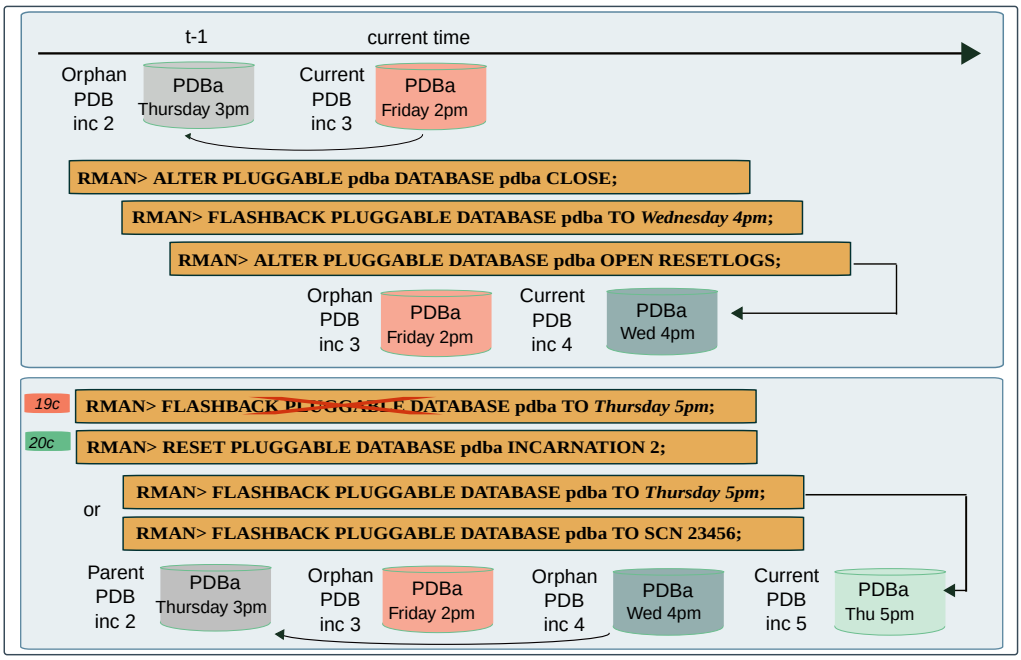
- Details: PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past
  This page provides more detailed information about new possibilities about PDB PITR or flashback to any time in the recent past.
- Practice: Flashbacking PDBs to Any Time in the Recent Past
  This practice shows how to perform a PDB PITR/Flashback to a specific time, then a PDB PITR/Flashback to a PDB time on an orphan PDB incarnation.

Related Topics

- *Oracle® Database Backup and Recovery User's Guide*

Details: PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past

This page provides more detailed information about new possibilities about PDB PITR or flashback to any time in the recent past.



Oracle database 20c allows point-in-time recovery or flashback of pluggable database to a point in time which lies on an orphaned PDB branch. This allows you to take back the database to anytime within a certain number of days and therefore rewind data back in time to correct any problems caused by logical data corruption or user errors:

- Allows PDB PITR/Flashback to any time as long as there is enough redo and flashback data and there is no CDB resetlogs
- Performs PDB point-in-time recovery/flashback to a PDB restore point on orphan PDB incarnation across multiple DB incarnation: A user can PITR/flashback a pluggable database to any point on a different database incarnation other than the current database incarnation as long as the database incarnation is on the current database ancestor path and sufficient redo/flashback data exists. Oracle does not support PDB PITR/Flashback to any point on orphaned database incarnation. The reason being, a user should be able to recover the CDB with one pass of media recovery after restoring any backup.

- Allows a DBA to issue a new RMAN command to set PDB incarnation before PDB PITR/Flashback to a SCN

Performing a flashback operation on a particular PDB modifies the data files for that PDB only. The remaining PDBs in the CDB are not impacted. The point in time to which the PDB must be flashed back is specified using a specific time, SCN, CDB restore point, PDB restore point, PDB clean restore point, or PDB guaranteed restore point.

Practice: Flashbacking PDBs to Any Time in the Recent Past

This practice shows how to perform a PDB PITR/Flashback to a specific time, then a PDB PITR/Flashback to a PDB time on an orphan PDB incarnation.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting flashbacking data in `PDB20`, execute the `/home/oracle/labs/M104782GC10/setup_Flashback.sh` shell script that enables flahsback in the CDB, recreates `PDB20` and creates the HR schema in `PDB20`.

```
$ cd /home/oracle/labs/M104782GC10
$ /home/oracle/labs/M104782GC10/setup_Flashback.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 13 11:05:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER DATABASE FLASHBACK on;

Database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 13 11:15:41 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER PLUGGABLE DATABASE pdb20 CLOSE;

Pluggable database altered.
```

```
SQL> ALTER SESSION SET db_create_file_dest='/home/oracle/labs';

Session altered.

SQL> DROP PLUGGABLE DATABASE pdb20 INCLUDING DATAFILES;

Pluggable database dropped.

SQL> CREATE PLUGGABLE DATABASE pdb20
  2      ADMIN USER pdb_admin IDENTIFIED BY password ROLES=(CONNECT)
  3      CREATE_FILE_DEST='/home/oracle/labs';

Pluggable database created.

SQL>
SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN;

Pluggable database altered.

SQL> exit


Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 13 11:05:14 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0


specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:


PL/SQL procedure successfully completed.

User created.

ALTER USER hr DEFAULT TABLESPACE users
...
Commit complete.

PL/SQL procedure successfully completed.
```

```
SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Connect to the CDB root and check that the CDB is open and enabled for flashback.

```
$ sqlplus / AS SYSDBA
SQL*Plus: Release 20.0.0.0.0 - Production on Fri Mar 13 07:10:40 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SELECT open_mode, flashback_on FROM v$database;

OPEN_MODE            FLASHBACK_ON
-------------------- ------------------
READ WRITE           YES

SQL>
```

4. Before any DDL or DML command is executed on `HR.EMPLOYEES` table in `PDB20`, display the current SCN, its associated timestamp and the incarnations of the PDB.

```
SQL> CONNECT sys@PDB20 AS SYSDBA
Enter password: password
Connected.
SQL> COL TIMESTAMP FORMAT A40
SQL> SELECT CURRENT_SCN, SCN_TO_TIMESTAMP(CURRENT_SCN) "TIMESTAMP" from
V$DATABASE;

CURRENT_SCN SCN_TO_TIMESTAMP(CURRENT_SCN)
-----------
----------------------------------------------------------------------
    3880324 13-MAR-20 07.12.24.000000000 AM

SQL> SELECT con_id, status, pdb_incarnation# inc#, begin_resetlogs_scn,
end_resetlogs_scn
     FROM   v$pdb_incarnation ORDER BY 3;

    CON_ID STATUS        INC# BEGIN_RESETLOGS_SCN END_RESETLOGS_SCN
---------- ------- ---------- ------------------ ----------------
         4 PARENT           0                  1                1
         4 CURRENT          0            2667602          2667602

SQL>
```

Possible ORPHAN incarnations would come from previous PDB resetlogs.

5. Display the number of rows in HR.EMPLOYEES table.

```
SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL>
```

6. A user makes an accidental removal of the HR.EMPLOYEES table in PDB20.

```
SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;

Table dropped.

SQL>
```

7. Flashback the PDB so as to restore the dropped table. Ensure that PDB20 is closed. Other PDBs can be open and operational.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL>
```

8. Flashback the data back to the point before the table was dropped. You need not set the orphan PDB incarnation if the flashback operation is to a specified time or restore point. Determine the desired SCN or point in time for the Flashback Database command. This point must be within the current CDB incarnation or an ancestor CDB incarnation.

```
SQL> FLASHBACK PLUGGABLE DATABASE TO SCN 3880324;

Flashback complete.

SQL>
```

9. Open PDB20 with RESETLOGS.

```
SQL> ALTER PLUGGABLE DATABASE OPEN RESETLOGS;

Pluggable database altered.

SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL>
```

10. Display the incarnations of PDB20.

```
SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn,
end_resetlogs_scn
                FROM v$pdb_incarnation ORDER BY 1, 2;
    CON_ID       INC# STATUS  INCARNATION_SCN END_RESETLOGS_SCN
---------- ---------- ------- -------------- -----------------
         4          0 PARENT          2667602           2667602
         4          1 CURRENT         3880344           3881083

SQL>
```

11. Increase the salary of the employees in HR.EMPLOYEES by 2 for some employees.

```
SQL> SELECT min(salary), MAX(salary) FROM hr.employees;

MIN(SALARY) MAX(SALARY)
----------- -----------
       2100       24000

SQL> UPDATE hr.employees SET salary=salary*2 WHERE employee_id<200;

100 rows updated.

SQL> COMMIT;

Commit complete.

SQL> SELECT CURRENT_SCN, SCN_TO_TIMESTAMP(CURRENT_SCN) "TIMESTAMP" from
V$DATABASE;

CURRENT_SCN TIMESTAMP
----------- ---------------------------------------
    3881391 13-MAR-20 07.16.33.000000000 AM

SQL>
```

12. Two minutes later, you delete the employee 206.

```
SQL> DELETE FROM hr.employees WHERE employee_id=206;

1 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       106

SQL> SELECT CURRENT_SCN, SCN_TO_TIMESTAMP(CURRENT_SCN) "TIMESTAMP" from
```

```
V$DATABASE;

CURRENT_SCN TIMESTAMP
----------- ---------------------------------------
    3882392 13-MAR-20 07.20.27.000000000 AM

SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn,
end_resetlogs_scn
               FROM v$pdb_incarnation ORDER BY 1, 2;

    CON_ID       INC# STATUS  INCARNATION_SCN END_RESETLOGS_SCN
---------- ---------- ------- --------------- -----------------
         4          0 PARENT          2667602           2667602
         4          1 CURRENT         3880344           3881083

SQL>
```

13. You decide to flashback the data back to the point before the table was dropped.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> FLASHBACK PLUGGABLE DATABASE TO SCN 3880324;

Flashback complete.

SQL> ALTER PLUGGABLE DATABASE OPEN RESETLOGS;

Pluggable database altered.

SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL> SELECT min(salary), MAX(salary) FROM hr.employees;

MIN(SALARY) MAX(SALARY)
----------- -----------
       2100       24000

SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn,
end_resetlogs_scn
     FROM v$pdb_incarnation ORDER BY 1, 2;
  2
    CON_ID       INC# STATUS  INCARNATION_SCN END_RESETLOGS_SCN
---------- ---------- ------- --------------- -----------------
         4          0 PARENT          2667602           2667602
         4          1 ORPHAN          3880344           3881083
         4          2 CURRENT         3880325           3882600

SQL>
```

14. Users ask for resetting PDB20 as it was after the salaries were updated and before the employee 206 was deleted. This state of PDB20 belongs to incarnation 1 of PDB20. Set the orphan PDB incarnation to which the flashback PDB operation must be performed. This step is required because the flashback operation is to an SCN or specific time in an orphan PDB incarnation.

```
SQL> RESET PLUGGABLE DATABASE TO INCARNATION 1;
SP2-0734: unknown command beginning "RESET PLUG..." - rest of line
ignored.
SQL> EXIT
$
```

This command exists only in RMAN.

```
$ rman TARGET sys@PDB20
target database Password: password
connected to target database: CDB20:PDB20 (DBID=2289122758)

RMAN> LIST INCARNATION OF PLUGGABLE DATABASE pdb20;

using target database control file instead of recovery catalog
List of Pluggable Database Incarnations
DB Key  PDB Key PDBInc Key DBInc Key   PDB Name   Status     Inc
SCN           Inc Time           Begin Reset SCN   Begin Reset Time
------- ------- --------   ---------   -------    --------
--------------   ------------------  ---------------
------------------
2       4        2          2          PDB20      CURRENT
3880325          13-MAR-20            3882600          13-MAR-20
End Reset SCN:3882600          End Reset Time:13-MAR-20
Guid:A0B8281946B32375E053424C960A082A
2       4        1          2          PDB20      ORPHAN
3880344          13-MAR-20            3881083          13-MAR-20
End Reset SCN:3881083          End Reset Time:13-MAR-20
Guid:A0B8281946B32375E053424C960A082A
2       4        0          2          PDB20      PARENT
2667602          12-MAR-20            2667602          12-MAR-20
End Reset SCN:2667602          End Reset Time:12-MAR-20
Guid:A0B8281946B32375E053424C960A082A

RMAN> RESET PLUGGABLE DATABASE pdb20 TO INCARNATION 1;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of reset database command at 03/13/2020 07:28:33
RMAN-05625: command not allowed when connected to a pluggable database

RMAN> exit
Recovery Manager complete.
$
$ rman TARGET /

Recovery Manager: Release 20.0.0.0.0 - Production on Mon Mar 13
11:50:04 2020
```

```
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: CDB20 (DBID=2732805675)

RMAN> ALTER PLUGGABLE DATABASE pdb20 CLOSE;

using target database control file instead of recovery catalog
Statement processed

RMAN> RESET PLUGGABLE DATABASE pdb20 TO INCARNATION 1;

pluggable database reset to incarnation 1

RMAN> FLASHBACK PLUGGABLE DATABASE pdb20 TO SCN 3880344;
Starting flashback at 13-JAN-20
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=148 device type=DISK

starting media recovery
media recovery failed
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of flashback command at 03/13/2020 07:31:00
ORA-39889: Specified System Change Number (SCN) or timestamp is in the
middle of a previous PDB RESETLOGS operation.

RMAN> exit
```

What does this error mean?

```
$ oerr ora 39889
39889, 00000, "Specified System Change Number (SCN) or timestamp is in
the middle of a previous PDB RESETLOGS operation."
// *Cause:  The specified System Change Number (SCN) or timestamp was
in the
//          middle of a previous PDB RESETLOGS operation. More
specifically,
//          each PDB RESETLOGS operation may create a PDB incarnation
as shown
//          in v$pdb_incarnation. Any SCN between INCARNATION_SCN and
//          END_RESETLOGS_SCN or any timestamp between INCARNATION_TIME
and
//          END_RESETLOGS_TIME as shown in v$pdb_incarnation is
considered in
//          the middle of the PDB RESETLOGS operation.
// *Action: Flashback the PDB to an SCN or timestamp that is not in the
middle
//          of a previous PDB RESETLOGS operation. If flashback to a
SCN on the
//          orphan PDB incarnation is required, then use
```

```
//          "RESET PLUGGABLE DATABASE TO INCARNATION" RMAN command to
specify
//          the pluggable database incarnation along which flashback to
the
//          specified SCN must be performed. Also, ensure that the
feature is
//          enabled.
$
```

Use the SCN displayed at the end of step 11.

```
$ rman TARGET /

Recovery Manager: Release 20.0.0.0.0 - Production on Mon Mar 13
11:50:04 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: CDB20 (DBID=2732805675)

RMAN> RESET PLUGGABLE DATABASE pdb20 TO INCARNATION 1;

pluggable database reset to incarnation 1

RMAN> FLASHBACK PLUGGABLE DATABASE pdb20 TO SCN 3881391;
Starting flashback at 13-MAR-20
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK


starting media recovery
archived log for thread 1 with sequence 9 is already on disk as
file /u03/app/oracle/fast_recovery_area/CDB20_IAD3CV/archivelog/
2020_04_07/o1_mf_1_9_h8s80s3f_.arc
archived log for thread 1 with sequence 10 is already on disk as
file /u03/app/oracle/fast_recovery_area/CDB20_IAD3CV/archivelog/
2020_04_07/o1_mf_1_10_h8s80t1w_.arc
archived log for thread 1 with sequence 11 is already on disk as
file /u03/app/oracle/fast_recovery_area/CDB20_IAD3CV/archivelog/
2020_04_07/o1_mf_1_11_h8s80y54_.arc

media recovery complete, elapsed time: 00:00:25

Finished flashback at 13-MAR-20

RMAN> EXIT

Recovery Manager complete.
$
```

15. Open the PDB and verify that the data is restored with the employees' salaries updated and the employee 206 restored too.

```
$ sqlplus sys@PDB20 AS SYSDBA
Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER PLUGGABLE DATABASE pdb20 OPEN RESETLOGS;

Pluggable database altered.

SQL> CONNECT system@PDB20
Enter password: password
Connected.
SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
       107

SQL> SELECT min(salary), MAX(salary) FROM hr.employees;

MIN(SALARY) MAX(SALARY)
----------- -----------
       4200       48000

SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn,
end_resetlogs_scn
        FROM v$pdb_incarnation ORDER BY 1, 2;

    CON_ID       INC# STATUS  INCARNATION_SCN END_RESETLOGS_SCN
---------- ---------- ------- --------------- -----------------
         4          0 PARENT          2667602           2667602
         4          1 PARENT          3880344           3881083
         4          2 ORPHAN          3880325           3882600
         4          3 CURRENT         3881392           3884391

SQL> EXIT
$
```

Autonomous Health Framework

- Oracle Trace File Analyzer Real-Time Health Summary
- Oracle Trace File Analyzer Log File Life Cycle Enhancements

Oracle Trace File Analyzer Real-Time Health Summary

Oracle Trace File Analyzer generates a real-time health summary report, which shows performance degradation due to faults and workload issues.

Similar to the status scorecard of the deployment configurations that Oracle ORAchk and Oracle EXAchk generate, Oracle Trace File Analyzer also provides a readily consumable and trackable scoring for operational status. The health summary consists of scores in the categories of availability, health, workload, and capacity broken down from cluster-wide through the database, instance, service, and hardware resource.

Related Topics

- *Oracle® Autonomous Health Framework User's Guide*

Oracle Trace File Analyzer Log File Life Cycle Enhancements

Oracle Trace File Analyzer archives log files before purging them upon each rotation.

The Oracle Database and Oracle Grid Infrastructure deployments generate a large number of logs and trace files. Oracle Trace File Analyzer does not archive these files. You have to create custom jobs if you need this history for support or auditing purposes. The enhancement in this release builds in the desired archiving functionality and thus removes the need for custom scripts.

Related Topics

- *Oracle® Autonomous Health Framework User's Guide*

Oracle Multitenant

- MAX_IDLE_BLOCKER_TIME Parameter
- Expanded Syntax for PDB Application Synchronization
- Details: Using non-CDBs and CDBs
  This page provides information about the availability of CDBs only in Oracle Database 20c. The non-CDB architecture was deprecated in Oracle Database 12c. It is desupported in Oracle Database 20c which means that the Oracle Universal Installer and DBCA can no longer be used to create non-CDB Oracle Database instances.

MAX_IDLE_BLOCKER_TIME Parameter

`MAX_IDLE_BLOCKER_TIME` sets the number of minutes that a session holding needed resources can be idle before it is a candidate for termination.

`MAX_IDLE_TIME` sets limits for all idle sessions, whereas `MAX_IDLE_BLOCKER_TIME` sets limits only for idle sessions consuming resources. `MAX_IDLE_TIME` can be problematic for a connection pool because it may continually try to re-create the sessions terminated by this parameter.

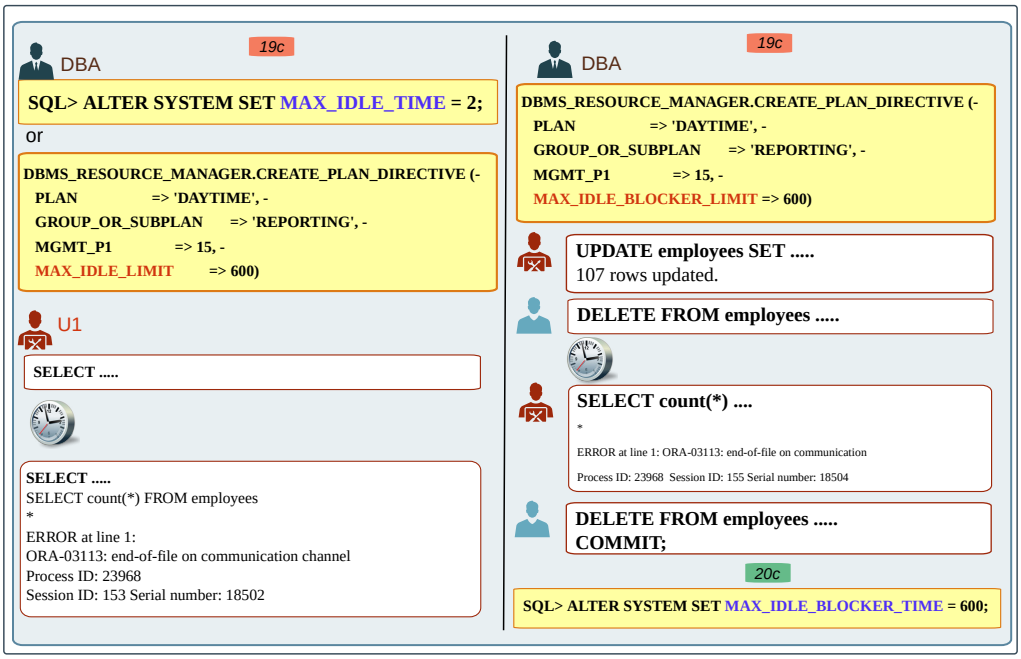
- Details: MAX_IDLE_BLOCKER_TIME Parameter
  This page provides more detailed information about the new initialization parameter `MAX_IDLE_BLOCKER_TIME` influencing sessions behavior.
- Practice: Using MAX_IDLE_BLOCKER_TIME Parameter
  This practice shows how to terminate a blocking session by using the new initialization parameter `MAX_IDLE_BLOCKER_TIME`.

Related Topics

- *Oracle® Multitenant Administrator's Guide*

Details: MAX_IDLE_BLOCKER_TIME Parameter

This page provides more detailed information about the new initialization parameter MAX_IDLE_BLOCKER_TIME influencing sessions behavior.



In Oracle Database 19c, you can specify an amount of time that a session can be idle, after which it is terminated. You can define the maximum session idle time, by setting:

- The MAX_IDLE_TIME resource plan directive, in seconds. Default is NULL, which implies unlimited.

```
SQL> EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (-
                    PLAN              => 'DAYTIME', -
                    GROUP_OR_SUBPLAN => 'REPORTING', -
                    MGMT_P1           => 15, -
                    MAX_IDLE_LIMIT    => 600)
```

- The MAX_IDLE_TIME initialization parmeter, in minutes. The default value of 0 indicates that there is no limit.

You can also specify a more stringent idle time limit that applies only to sessions that are idle consuming resources and therefore blocking other sessions, by setting the MAX_IDLE_BLOCKER_TIME resource plan directive that indicates the maximum session idle time of a blocking session. Default is NULL, which implies unlimited.

Oracle Database 20c allows you to set the MAX_IDLE_BLOCKER_TIME initialization parmeter to define the maximum session idle time of a blocking session, in minutes. The default value of 0 indicates that there is no limit.

Practice: Using MAX_IDLE_BLOCKER_TIME Parameter

This practice shows how to terminate a blocking session by using the new initialization parameter MAX_IDLE_BLOCKER_TIME.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Prepare two terminal sessions, one logged in PDB20 as HR and another one logged in PDB20 as SYSTEM.
   a. Log in PDB20 as SYSTEM.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 20 08:20:09
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> SET SQLPROMPT "SQL system> "
SQL system>
```

   b. Log in PDB20 as HR.

```
$ sqlplus hr@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Jan 20 08:20:09
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> SET SQLPROMPT "SQL hr> "
SQL hr>
```

3. In the SYSTEM session, set the initialization parameter MAX_IDLE_BLOCKER_TIME to two minutes.

```
SQL system> ALTER SYSTEM SET max_idle_blocker_time=2;

System altered.

SQL system> SHOW PARAMETER max_idle_blocker_time

NAME                                 TYPE        VALUE
```

```
------------------------------ -----------
------------------------------
max_idle_blocker_time          integer     2
SQL system>
```

4. In the HR session, update the employees' salary.

```
SQL hr> UPDATE hr.employees SET salary=salary*2;

107 rows updated.

SQL hr>
```

5. In the SYSTEM session, set all employees' commission percentage to 0. The statement waits for the lock resources held on the row by HR be released.

```
SQL system> UPDATE hr.employees SET commission_pct=0;
```

After two minutes, observe that the statement is executed.

```
107 rows updated.

SQL system>
```

6. Back in the HR session, query the result of the salaries update.

```
SQL hr> SELECT salary FROM hr.employees;
SELECT salary FROM hr.employees       *
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 32314
Session ID: 274 Serial number: 8179

SQL hr> EXIT
$
```

The session was automatically terminated because it held resource for a duration longer than two minutes.

Observe the DIAG trace file:

```
$ cd /u01/app/oracle/diag/rdbms/database_unq_name/database_name/trace
$ ls -ltr
...
-rw-r----- 1 oracle oinstall    4139 Mar 16 04:38
CDB20_dia0_30961_base_1.trm
-rw-r----- 1 oracle oinstall   16101 Mar 16 04:38
CDB20_dia0_30961_base_1.trc
-rw-r----- 1 oracle oinstall    1067 Mar 16 04:48 CDB20_mmon_31003.trm
-rw-r----- 1 oracle oinstall    2614 Mar 16 04:48 CDB20_mmon_31003.trc
-rw-r----- 1 oracle oinstall    1107 Mar 16 04:49 CDB20_dbrm_30949.trm
-rw-r----- 1 oracle oinstall    3398 Mar 16 04:49 CDB20_dbrm_30949.trc
...
```

```
$ cat CDB20_dia0_30961_base_1.trc
...
HM: Session with ID 274 serial # 8179 (U01I) on single instance 1 is
hung
    and is waiting on 'SQL*Net message from client' for 96 seconds.
    Session was previously waiting on 'SQL*Net more data to client'.
    Session ID 274 is blocking 1 session
...
HM: Session with ID 136 serial # 42403 (U011) on single instance 1 is
hung
    and is waiting on 'enq: TX - row lock contention' for 96 seconds.
    Session was previously waiting on 'db file sequential read'.
    Final Blocker is Session ID 274 serial# 8179 on instance 1
     which is waiting on 'SQL*Net message from client' for 108 seconds
     p1: 'driver id'=0x54435000, p2: '#bytes'=0x1, p3: ''=0x0
...
*** 2020-03-16T04:31:35.031598+00:00 (CDB$ROOT(1))
All Current Hang Statistics

                     current number of hangs 1
    hangs:current number of impacted sessions 2
                  current number of deadlocks 0
deadlocks:current number of impacted sessions 0
                 current number of singletons 0
      current number of local active sessions 2
        current number of local hung sessions 1

Suspected Hangs in the System and possibly Rebuilt Hangs
                     Root       Chain Total               Hang
  Hang Hang          Inst Root  #hung #hung  Hang   Hang  Resolution
    ID Type Status   Num  Sess   Sess  Sess  Conf   Span  Action
 ----- ---- -------- ---- ----- ----- ----- ------ ------
-------------------
     1 HANG    VALID    1   274     2     2    LOW  LOCAL Terminate
Process

  Inst  Sess   Ser             Proc  Wait    Wait
   Num    ID    Num      OSPID  Name Time(s) Event
  ----- ------ ----- --------- ----- ------- -----
        PDBID PDBNm
        ----- ---------------
      1    136 42403     32583  U011      97 enq: TX - row lock
contention
            7 PDB20
      1    274  8179     32314  U01I     110 SQL*Net message from client
            7 PDB20
;..
HM: current SQL: UPDATE hr.employees SET commission_pct=0

                                                        IO
 Total  Self-         Total  Total  Outlr  Outlr  Outlr
  Hung  Rslvd  Rslvd   Wait WaitTm   Wait WaitTm   Wait
  Sess  Hangs  Hangs  Count   Secs  Count   Secs  Count Wait Event
------ ------ ------ ------ ------ ------ ------ ------ -----------
     1      0      0      0      0      0      0      0 enq: TX - row
```

```
lock contention
...
HM: current SQL: UPDATE employees SET salary=salary*2
...
HM: Session ID 274 serial# 8179 ospid 32314 on instance 1 in Hang ID 1
    was considered hung but is now no longer hung

HM: Session with ID 274 with serial number 8179 is no longer hung

*** 2020-03-16T04:38:25.114410+00:00 (CDB$ROOT(1))
HM: Hang ID=1 detected at 03/16/2020 04:31:34 with victim:1/274/8179
    Evt:'SQL*Net message from client', SELF-RESOLVED after 0 matches
(0) (1).
$
```

You can also read the PMON trace file.

```
$ cat /u01/app/oracle/diag/rdbms/cdb20/CDB20/trace/CDB20_pmon_30913.trc
...
Kill idle blocker, hang detected

*** 2020-03-16T04:32:04.240685+00:00 ((7))
Idle session sniped info:
reason=max_idle_blocker_time parameter sess=0x86a06a20 sid=274
serial=8179 idle=2 limit=2 event=SQL*Net message from client
client details:
  O/S info: user: oracle, term: pts/0, ospid: 32312
  machine: edcdr8p1 program: sqlplus@edcdr8p1 (TNS V1-V3)
  application name: SQL*Plus, hash value=3669949024
Current SQL:
UPDATE employees SET salary=salary*2
End of Idle session sniped info
KILL SESSION for sid=(274, 8179):
  Reason = max_idle_blocker_time parameter, idle time = 2 mins,
currently waiting on 'SQL*Net message from
...
$
```

7. In the SYSTEM session, query the employees' commission percentage.

```
SQL system> SELECT DISTINCT commission_pct FROM hr.employees;

COMMISSION_PCT
--------------
             0

SQL system> EXIT
$
```

Expanded Syntax for PDB Application Synchronization

The `ALTER PLUGGABLE DATABASE APPLICATION ... SYNC` statement now accepts multiple application names and names to be excluded. For example, a single

statement issued in an application PDB can synchronize `app1` and `app2`, or synchronize all applications except `app3`.

The expanded syntax enables you to reduce the number of synchronization statements. Also, the database replays the statements in correct order. Assume that you upgrade `ussales` from v1 to v2, and then upgrade `eusales` from v1 to v2, and then upgrade `ussales` from v2 to v3. The statement `ALTER PLUGGABLE DATABASE APPLICATION ussales, eusales SYNC` replays the statements in sequence, upgrading `ussales` to v2, then `eusales` to v2, and then `ussales` to v3.

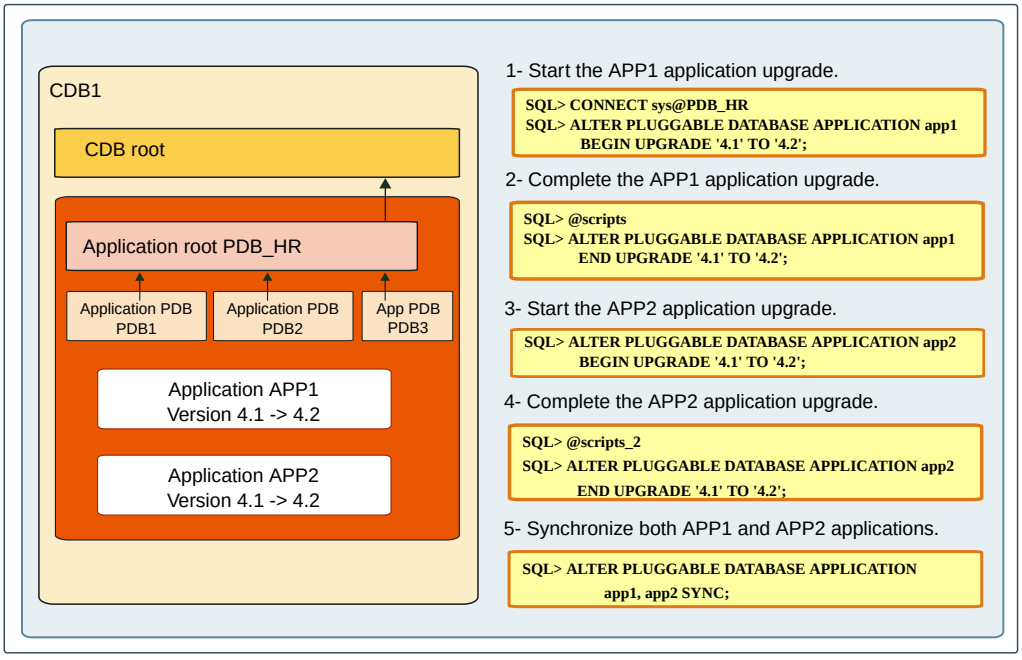
- Details: Expanded Syntax for PDB Application Synchronization
  This page provides more detailed information about enhancement of applications synchronization in application PDBs.
- Practice: Synchronizing Multiple Applications In Application PDBs
  This practice shows how to reduce the number of synchronization statements when you have to synchronize multiple applications in application PDBs. In previous Oracle Database versions, you had to execute as many synchronization statements as applications.

Related Topics

- *Oracle® Multitenant Administrator's Guide*

Details: Expanded Syntax for PDB Application Synchronization

This page provides more detailed information about enhancement of applications synchronization in application PDBs.



In Oracle Database 19c, the `ALTER PLUGGABLE DATABASE APPLICATION … SYNC` statement accepts only one application name to synchronize with the application root. You had to execute the statement as many times as the number of applications to synchronize with the application root.

In Oracle Database 20c, the benefit of the `ALTER PLUGGABLE DATABASE APPLICATION … SYNC` statement is that it allows you to execute the statement only once for multiple

application names. For example, a single statement issued in an application PDB can synchronize `apexapp` and `ordsapp`, or synchronize all applications except `ordsapp`.

When applications depend on one another, synchronizing them in a single statement is necessary for functional correctness. Assume that you upgrade `apexapp` from 1.0 to 2.0, upgrade `ordsapp` from 1.0 to 2.0, and then upgrade `apexapp` to 3.0. The statement `ALTER PLUGGABLE DATABASE APPLICATION apexapp, ordsapp SYNC` replays the upgrades in sequence, upgrading `apexapp` to 2.0, `ordsapp` to 2.0, and then `apexapp` to 3.0. Synchronizing `apexapp` and then `ordsapp` in separate statements does not preserve the upgrade order.

Practice: Synchronizing Multiple Applications In Application PDBs

This practice shows how to reduce the number of synchronization statements when you have to synchronize multiple applications in application PDBs. In previous Oracle Database versions, you had to execute as many synchronization statements as applications.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Use the `/home/oracle/labs/M104780GC10/setup_apps.sh` shell script to install the `TOYS_APP` and the `SALES_TOYS_APP` applications in the `TOYS_ROOT` application container for both `ROBOTS` and `DOLLS` application PDBs. The script defines the application container, installs the two applications in the application container, and finally creates the two application PDBs in the application container.
   a. To be able to connect during the shell script execution to `TOYS_ROOT`, `ROBOTS` and `DOLLS`, create the entries in the `tnsnames.ora` file as explained in practices environment.
   b. Execute the shell script.

```
$ cd /home/oracle/labs/M104780GC10
$ /home/oracle/labs/M104780GC10/setup_apps.sh
...
SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE IMMEDIATE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE robots INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE dolls INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE toys_root INCLUDING DATAFILES;

Pluggable database dropped.

SQL> ALTER SESSION SET db_create_file_dest='/home/oracle/labs/
toys_root';

Session altered.
```

```
SQL> CREATE PLUGGABLE DATABASE toys_root AS APPLICATION CONTAINER
  2    ADMIN USER admin IDENTIFIED BY password  ROLES=(CONNECT);

Pluggable database created.

SQL> alter PLUGGABLE DATABASE toys_root open;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Nov 29 03:03:18
2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app begin install
'1.0';

Pluggable database altered.

SQL>
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/home/oracle/labs/
toys_root';

System altered.

SQL> CREATE TABLESPACE toys_tbs DATAFILE SIZE 100M autoextend on
next 10M maxsize 200M;

Tablespace created.

SQL> create user toys_owner identified by password container=all;

User created.

SQL> grant create session, dba to toys_owner;

Grant succeeded.

SQL>
SQL> CREATE TABLE toys_owner.categories SHARING=DATA (c1 number,
category varchar2(20));

Table created.
```

```
SQL> INSERT INTO toys_owner.categories VALUES (1,'GAMES');

1 row created.

SQL> INSERT INTO toys_owner.categories VALUES (2,'PUPPETS');

1 row created.

SQL> INSERT INTO toys_owner.categories VALUES (3,'VEHICLES');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app end install
'1.0';

Pluggable database altered.

SQL>
SQL> ALTER PLUGGABLE DATABASE APPLICATION sales_toys_app BEGIN
INSTALL '1.0';

Pluggable database altered.

SQL>
SQL> CREATE USER sales_toys IDENTIFIED BY password CONTAINER=ALL;

User created.

SQL> GRANT create session, dba TO sales_toys;

Grant succeeded.

SQL> ALTER USER sales_toys DEFAULT TABLESPACE toys_tbs;

User altered.

SQL> CREATE TABLE sales_toys.sales_data sharing=extended data
  2  (year       number(4),
  3   region     varchar2(10),
  4   quarter    varchar2(4),
  5   revenue    number);

Table created.

SQL> INSERT INTO sales_toys.sales_data VALUES (2019,'US','Q1',
100000);

1 row created.
```

```
SQL> INSERT INTO sales_toys.sales_data VALUES (2019,'US','Q2',
400000);

1 row created.

SQL> INSERT INTO sales_toys.sales_data VALUES (2019,'EU','Q2',
50000);

1 row created.

SQL> INSERT INTO sales_toys.sales_data VALUES (2019,'ASIA','Q3',
300000);

1 row created.

SQL> INSERT INTO sales_toys.sales_data VALUES (2019,'EU','Q3',
20000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
SQL> ALTER PLUGGABLE DATABASE APPLICATION sales_toys_app END
INSTALL '1.0';

Pluggable database altered.

SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Nov 29 03:03:37
2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL>
SQL> ALTER SESSION SET DB_CREATE_FILE_DEST='/home/oracle/labs/
toys_root/robots';

Session altered.

SQL> CREATE PLUGGABLE DATABASE robots ADMIN USER admin identified
```

```
by password ROLES=(CONNECT);

Pluggable database created.

SQL> ALTER SESSION SET DB_CREATE_FILE_DEST='/home/oracle/labs/
toys_root/dolls';

Session altered.

SQL> CREATE PLUGGABLE DATABASE dolls ADMIN USER admin identified by
password ROLES=(CONNECT);

Pluggable database created.

SQL>
SQL> alter pluggable database robots open;

Pluggable database altered.

SQL> alter pluggable database dolls open;

Pluggable database altered.

SQL>
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Display the applications installed in the application container.

```
$ sqlplus / AS SYSDBA

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 05:29:42 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL> COL app_name FORMAT A16
SQL> COL app_version FORMAT A12
SQL> COL pdb_name FORMAT A10
SQL> SELECT app_name, app_version, app_status, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  a.con_id = p.pdb_id
     AND    app_name NOT LIKE '%APP$%'
     ORDER BY 1;

APP_NAME         APP_VERSION  APP_STATUS   PDB_NAME
---------------- ------------ ------------ ----------
SALES_TOYS_APP   1.0          NORMAL       TOYS_ROOT
TOYS_APP         1.0          NORMAL       TOYS_ROOT
```

```
SQL>
```

Observe that the applications `toys_app` and `sales_toys_app` are installed in the application container at version 1.0.

4. Synchronize the application PDBs with the new applications `toys_app` and `sales_toys_app` installed.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: password
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app, sales_toys_app SYNC;

Pluggable database altered.

SQL> SELECT app_name, app_version, app_status, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  a.con_id = p.pdb_id
     AND    app_name NOT LIKE '%APP$%'
     ORDER BY 1;

APP_NAME         APP_VERSION  APP_STATUS   PDB_NAME
---------------- ------------ ------------ ----------
SALES_TOYS_APP   1.0          NORMAL       ROBOTS
TOYS_APP         1.0          NORMAL       ROBOTS

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: password
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app, sales_toys_app SYNC;

Pluggable database altered.

SQL> SELECT app_name, app_version, app_status, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  a.con_id = p.pdb_id
     AND    app_name NOT LIKE '%APP$%'
     ORDER BY 1;

APP_NAME         APP_VERSION  APP_STATUS   PDB_NAME
---------------- ------------ ------------ ----------
SALES_TOYS_APP   1.0          NORMAL       DOLLS
TOYS_APP         1.0          NORMAL       DOLLS

SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT app_name, app_version, app_status, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  a.con_id = p.pdb_id
     AND    app_name NOT LIKE '%APP$%'
     ORDER BY 1;

APP_NAME         APP_VERSION  APP_STATUS   PDB_NAME
---------------- ------------ ------------ ----------
SALES_TOYS_APP   1.0          NORMAL       DOLLS
SALES_TOYS_APP   1.0          NORMAL       ROBOTS
```

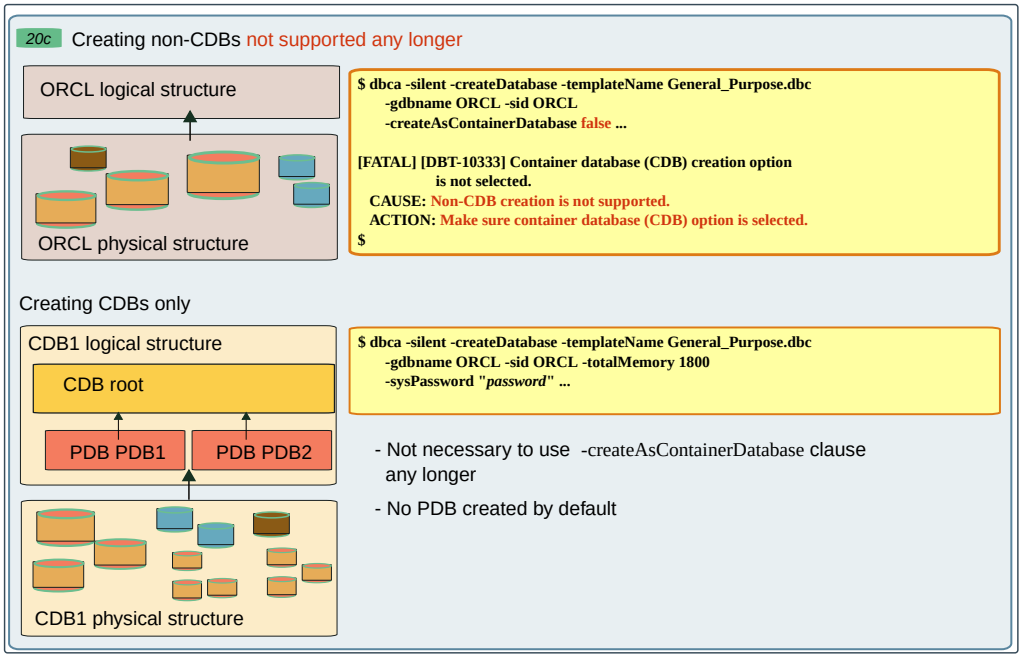
```
SALES_TOYS_APP    1.0          NORMAL       TOYS_ROOT
TOYS_APP          1.0          NORMAL       DOLLS
TOYS_APP          1.0          NORMAL       TOYS_ROOT
TOYS_APP          1.0          NORMAL       ROBOTS

6 rows selected.

SQL> EXIT
$
```

Details: Using non-CDBs and CDBs

This page provides information about the availability of CDBs only in Oracle Database 20c. The non-CDB architecture was deprecated in Oracle Database 12c. It is desupported in Oracle Database 20c which means that the Oracle Universal Installer and DBCA can no longer be used to create non-CDB Oracle Database instances.



A multitenant container database is the only supported architecture in Oracle Database 20c.

Tools and Languages

- Analytical SQL and Statistical Functions
- SQL

Analytical SQL and Statistical Functions

- Bitwise Aggregate Functions
- New Analytical and Statistical Aggregate Functions
- Enhanced Analytic Functions

Bitwise Aggregate Functions

New aggregate functions BIT_AND_AGG, BIT_OR_AGG, and BIT_XOR_AGG enable bitwise aggregation of integer columns and columns that can be converted or rounded to integer values.

Bitwise aggregation functions enable bitwise type processing directly in SQL. Use of these new functions improves overall query performance by eliminating unnecessary data movement and by taking full advantage of other database capabilities such as parallel processing.

- Practice: Using Bitwise Aggregate Functions
  This practice shows how to use the new BIT_AND_AGG, BIT_OR_AGG and BIT_XOR_AGG bitwise aggregate functions at the bit level of records within a group. BIT_AND_AGG, BIT_OR_AGG and BIT_XOR_AGG return the result of bitwise AND, OR and XOR operations respectively. These aggregates can be performed on a single numeric column or an expression. The return type of a bitwise aggregate operation is always a number.

Related Topics

- *Oracle® Database Data Warehousing Guide*

Practice: Using Bitwise Aggregate Functions

This practice shows how to use the new BIT_AND_AGG, BIT_OR_AGG and BIT_XOR_AGG bitwise aggregate functions at the bit level of records within a group. BIT_AND_AGG, BIT_OR_AGG and BIT_XOR_AGG return the result of bitwise AND, OR and XOR operations respectively. These aggregates can be performed on a single numeric column or an expression. The return type of a bitwise aggregate operation is always a number.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Connect to PDB20 as SYSTEM to query values with numbers and bitwise aggregate functions.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 08:48:55 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Mon Mar 16 2020 04:28:54 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
```

3. A bitwise AND is a binary operation that takes two equal-length binary representations and performs the logical AND operation on each pair of the corresponding bits. If both bits in the compared position are 1, the bit in the resulting binary representation is 1, otherwise, the result is 0. Apply the

BIT_AND_AGG function on two numbers. The bit pattern for the values used in the examples below are 01 for 1, 10 for 2, and 11 for 3.

```
SQL> WITH x AS (SELECT 2 c1 FROM dual UNION ALL SELECT 3 FROM dual)
              SELECT BIT_AND_AGG(c1) FROM x;

BIT_AND_AGG(C1)
---------------
              2

SQL>
```

4. A bitwise OR is a binary operation that takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 0 if both bits are 0, otherwise the result is 1. Apply the BIT_OR_AGG function on two numbers.

```
SQL> WITH x AS (SELECT 2 c1 FROM dual UNION ALL SELECT 3 FROM dual)
              SELECT BIT_OR_AGG(c1) FROM x;

BIT_OR_AGG(C1)
--------------
             3

SQL>
```

5. A bitwise XOR is a binary operation that takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. Therefore, the comparison of two bits results in 1 if the two bits are different, and 0 if they are equal. Apply the BIT_XOR_AGG function on two numbers.

```
SQL> WITH x AS (SELECT 2 c1 FROM dual UNION ALL SELECT 3 FROM dual)
              SELECT BIT_XOR_AGG(c1) FROM x;

BIT_XOR_AGG(C1)
---------------
              1

SQL> EXIT
$
```

New Analytical and Statistical Aggregate Functions

New analytical and statistical aggregate functions are available in SQL:

CHECKSUM computes the checksum of the input values or expression.

KURTOSIS functions KURTOSIS_POP and KURTOSIS_SAMP measure the tailedness of a data set where a higher value means more of the variance within the data set is the result of infrequent extreme deviations as opposed to frequent modestly sized deviations. Note that a normal distribution has a kurtosis of zero.

SKEWNESS functions SKEWNESS_POP and SKEWNESS_SAMP are measures of asymmetry in data. A positive skewness is means the data skews to the right of the center point. A negative skewness means the data skews to the left.

All of these new aggregate functions support the keywords ALL, DISTINCT, and UNIQUE.

With these additional SQL aggregation functions, you can write more efficient code and benefit from faster in-database processing.

- Practice: Detecting Data Tampering with the CHECKSUM Function
  This practice shows how to use the CHECKSUM aggregate function to detect changes in a table. The function can be applied on a column, a constant, a bind variable, or an expression involving them. All datatypes except ADT and JSON are supported. The order of the rows in the table does not affect the result.
- Practice: Measuring Asymmetry in Data with the SKEWNESS Functions
  This practice shows how to use the SKEWNESS_POP and SKEWNESS_SAMP aggregate functions to measure asymmetry in data. For a given set of values, the result of population skewness (SKEWNESS_POP) and sample skewness (SKEWNESS_SAMP) are always deterministic.
- Practice: Measuring Tailedness of Data with the KURTOSIS Functions
  This practice shows how to use the KURTOSIS_POP and KURTOSIS_SAMP aggregate functions to measure tailedness of data. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations. A normal distribution has a kurtosis of zero.

Related Topics

- *Oracle® Database Data Warehousing Guide*

Practice: Detecting Data Tampering with the CHECKSUM Function

This practice shows how to use the CHECKSUM aggregate function to detect changes in a table. The function can be applied on a column, a constant, a bind variable, or an expression involving them. All datatypes except ADT and JSON are supported. The order of the rows in the table does not affect the result.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Execute the /home/oracle/labs/M104784GC10/setup_SH_tables.sh shell script to create and load SH.SALES and SH.TIMES tables.

```
$ cd /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/setup_SH_tables.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Mar 25 03:18:51 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Wed Mar 25 2020 03:17:43 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
```

```
Tablespace dropped.

Tablespace created.

Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Mar 25 03:19:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Wed Mar 25 2020 03:18:51 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

specify password for SH as parameter 1:

specify default tablespace for SH as parameter 2:

specify temporary tablespace for SH as parameter 3:

specify password for SYS as parameter 4:

specify directory path for the data files as parameter 5:

writeable directory path for the log files as parameter 6:

specify version as parameter 7:

specify connect string as parameter 8:

Session altered.

User dropped.
...
loading TIMES using:
/home/oracle/labs/M104784GC10/sales_history/time_v3.ctl
/home/oracle/labs/M104784GC10/sales_history/time_v3.dat
/home/oracle/labs/M104784GC10/time_v3.log

SQL*Loader: Release 20.0.0.0.0 - Production on Wed Mar 25 03:10:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.

Path used:      Direct
Save data point reached - logical record count 1000.

Load completed - logical record count 1826.
```

```
Table TIMES:
  1826 Rows successfully loaded.
...
loading additonal SALES using:
/home/oracle/labs/M104784GC10/sales_history/dmsal_v3.ctl
/home/oracle/labs/M104784GC10/sales_history/dmsal_v3.dat
/home/oracle/labs/M104784GC10/dmsal_v3.log

SQL*Loader: Release 20.0.0.0.0 - Production on Wed Mar 25 03:10:45 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.

Path used:      Direct
Save data point reached - logical record count 100.
Save data point reached - logical record count 200.
Save data point reached - logical record count 300.
Save data point reached - logical record count 400.
Save data point reached - logical record count 500.
Save data point reached - logical record count 600.
Save data point reached - logical record count 700.
Save data point reached - logical record count 800.
Save data point reached - logical record count 900.
Save data point reached - logical record count 1000.
Save data point reached - logical record count 1100.
Save data point reached - logical record count 1200.
Save data point reached - logical record count 1300.
Save data point reached - logical record count 1400.
Save data point reached - logical record count 1500.
SQL*Loader-2026: the load was aborted because SQL Loader cannot
continue.

Load completed - logical record count 1600.

Table SALES:
  1500 Rows successfully loaded.
...
gathering statistics ...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL>
```

3. At the end of each month and fiscal period, for legislative reasons, there is an audit table that stores what was sold. Verify the amount sold at the end of fiscal year 1998.

```
SQL> CONNECT system@PDB20
Enter password: password
SQL> SET PAGES 100
SQL> SELECT amount_sold FROM sh.sales s
```

```
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

AMOUNT_SOLD
-----------
      22.99
      44.99
       7.99
     149.99
...
      11.99
      44.99
      49.99
      11.99
      44.99
      27.99
     149.99
      44.99

12400 rows selected.

SQL>
```

4. Before storing the data for auditing, note the `CHECKSUM` value. This will help you ensure that no one is tampering with old sales.

```
SQL> SELECT CHECKSUM(amount_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(AMOUNT_SOLD)
---------------------
               793409

SQL>
```

5. Meanwhile in another terminal session, called *SH* session, someone executes a batch that updates the amount sold.

```
$ /home/oracle/labs/M104784GC10/app_SH_tables.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Mar 25 03:28:37 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Last Successful login time: Wed Mar 25 2020 03:20:17 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

525 rows updated.

Commit complete.
```

```
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

6. In the initial terminal session, check that no one tampered with old sales.

```
SQL> SELECT CHECKSUM(amount_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(AMOUNT_SOLD)
---------------------
               835564

SQL>
```

   Since the checksum value is different from the value retrieved in step 4, someone tampered the data.

7. What happens if someone attempted to tamper with old sales? In the *SH* session, update some old sales but then rolls the transaction back.

```
$ sqlplus sh@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Mar 25 03:45:09 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Wed Mar 25 2020 03:28:37 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> UPDATE sh.sales SET amount_sold = amount_sold*2 WHERE time_id='30-
NOV-98';

525 rows updated.

SQL> ROLLBACK;

Rollback complete.

SQL>
```

8. In the initial terminal session, check that no one tampered with old sales.

```
SQL> SELECT CHECKSUM(amount_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(AMOUNT_SOLD)
```

```
--------------------
              835564

SQL>
```

The checksum value for the column is still the same as it was before the rolled back update.

9. Verify also the quantity sold at the end of fiscal year 1998 and the checksum value.

```
SQL> SELECT DISTINCT quantity_sold FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

QUANTITY_SOLD
-------------
            1

SQL>
```

As you can see, the quantity sold for any sales is one.

```
SQL> SELECT CHECKSUM(quantity_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(QUANTITY_SOLD)
-----------------------
                     0

SQL>
```

The checksum value is 0 which is not a distinguishable value from another quantity value.

What if you use the `DISTINCT` (or `UNIQUE`- `UNIQUE` is an Oracle specific keyword and not an ANSI standard)?

```
SQL> SELECT CHECKSUM(DISTINCT quantity_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(DISTINCTQUANTITY_SOLD)
-------------------------------
                         863352

SQL>
```

10. In the *SH* session, double the quantity for all sales.

```
SQL> UPDATE sh.sales SET quantity_sold = 2;

918843 rows updated.
```

```
SQL> COMMIT;

Commit complete.

SQL>
```

11. In the initial terminal session, check that no one tampered with old sales.

```
SQL> SELECT CHECKSUM(quantity_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(AMOUNT_SOLD)
---------------------
                    0

SQL>
```

The checksum value for the column is still the same as it was before the committed update.

```
SQL> SELECT CHECKSUM(DISTINCT quantity_sold) FROM sh.sales s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998;

CHECKSUM(DISTINCTQUANTITY_SOLD)
-------------------------------
                          65515

SQL>
```

The checksum value for the column is different from the one retrieved in step 9.

12. How is NULL considered? Still in the initial terminal session, check that no one tampered with old sales of the end of fiscal year 1998, stored in the SALES_TRANSACTIONS_EXT table whose amount sold is 1282.7.

    a. First, get the checksum value of old sales of the end of fiscal year 1998 whose amount sold is 1282.7.

```
SQL> SELECT CHECKSUM(DISTINCT amount_sold), CHECKSUM(DISTINCT
quantity_sold) FROM sh.SALES_TRANSACTIONS_EXT s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998 AND
amount_sold = to_number('1282.7');

CHECKSUM(AMOUNT_SOLD) CHECKSUM(QUANTITY_SOLD)
--------------------- ----------------------
               422955                 863352

SQL>
```

b. In the *SH* session, the user launched a batch that replaces the quantity sold by a null value for the old sales of the end of fiscal year 1998 whose amount sold is 1282.7.

```
SQL> @/home/oracle/labs/M104784GC10/batch.sql
...
SQL> EXIT
$
```

c. In the initial terminal session, get the new checksum value of old sales of the end of fiscal year 1998 whose amount sold is 1282.7 after the update.

```
SQL> SELECT CHECKSUM(DISTINCT amount_sold), CHECKSUM(DISTINCT
quantity_sold) FROM sh.SALES_TRANSACTIONS_EXT s
     JOIN sh.times t ON (s.time_id = t.time_id)
     WHERE fiscal_month_number = 12 AND fiscal_year = 1998 AND
amount_sold = to_number('1282.7');

CHECKSUM(AMOUNT_SOLD) CHECKSUM(QUANTITY_SOLD)
--------------------- ----------------------
               422955                 863352

SQL> SELECT amount_sold, quantity_sold FROM
sh.SALES_TRANSACTIONS_EXT
     WHERE  amount_sold = to_number('1282.7')
     AND quantity_sold IS NULL;

AMOUNT_SOLD QUANTITY_SOLD
----------- -------------
     1282.7

SQL> EXIT
$
```

Be aware that NULL values in `CHECKSUM` column are ignored.

Practice: Measuring Asymmetry in Data with the `SKEWNESS` Functions

This practice shows how to use the `SKEWNESS_POP` and `SKEWNESS_SAMP` aggregate functions to measure asymmetry in data. For a given set of values, the result of population skewness (`SKEWNESS_POP`) and sample skewness (`SKEWNESS_SAMP`) are always deterministic.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Connect to `PDB20` as `HR` and execute the `/home/oracle/labs/M104784GC10/Houses_Prices.sql` SQL script. The SQL script creates a table with skewed data.

```
$ cd /home/oracle/labs/M104784GC10
$ sqlplus hr@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 09:27:03 2020
Version 20.2.0.0.0
```

```
Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Mon Mar 16 2020 08:49:41 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> @/home/oracle/labs/M104784GC10/Houses_Prices.sql
SQL> SET ECHO ON
SQL>SQL> DROP TABLE houses;
DROP TABLE houses
           *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CREATE TABLE houses (house NUMBER, price_big_city NUMBER,
price_small_city NUMBER, price_date DATE);

Table created.

SQL> INSERT INTO houses VALUES (1,100000,10000, sysdate);

1 row created.

SQL> INSERT INTO houses VALUES (1,200000,15000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (1,300000,25000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (1,400000,28000, sysdate+2);

1 row created.

SQL> INSERT INTO houses VALUES (1,500000,30000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (1,600000,32000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (1,700000,35000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (1,800000,38000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (1,900000,40000, sysdate+5);
```

```
1 row created.

SQL>
SQL> INSERT INTO houses VALUES (2,2000000,1000000, sysdate+6);

1 row created.

SQL> INSERT INTO houses VALUES (2,200000,20000, sysdate);

1 row created.

SQL> INSERT INTO houses VALUES (2,400000,35000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (2,600000,55000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (2,800000,48000, sysdate+2);

1 row created.

SQL>
SQL> INSERT INTO houses VALUES (3,400000,40000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (3,500000,42000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (3,600000,45000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (3,700000,48000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (3,800000,49000, sysdate+5);

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

3. Display the table rows. The `HOUSE` column values refer to types of house that you want to look at and categorize the data that you look at statistically and compare with each other. With Skewness, you measure whether there is more data towards

the left or the right end of the tail (positive/negative) or how close you are to a normal distribution (skewness = 0).

```
SQL> SET PAGES 100
SQL> SELECT * FROM houses;

     HOUSE PRICE_BIG_CITY PRICE_SMALL_CITY PRICE_DAT
---------- -------------- ---------------- ---------
         1         100000            10000 05-FEB-20
         1         200000            15000 06-FEB-20
         1         300000            25000 06-FEB-20
         1         400000            28000 07-FEB-20
         1         500000            30000 08-FEB-20
         1         600000            32000 08-FEB-20
         1         700000            35000 09-FEB-20
         1         800000            38000 09-FEB-20
         1         900000            40000 10-FEB-20
         2        2000000          1000000 11-FEB-20
         2         200000            20000 05-FEB-20
         2         400000            35000 06-FEB-20
         2         600000            55000 06-FEB-20
         2         800000            48000 07-FEB-20
         3         400000            40000 08-FEB-20
         3         500000            42000 08-FEB-20
         3         600000            45000 09-FEB-20
         3         700000            48000 09-FEB-20
         3         800000            49000 10-FEB-20

19 rows selected.

SQL>
```

4. Display the result of population skewness prices (SKEWNESS_POP) and sample skewness prices (SKEWNESS_SAMP) for the three houses in the table.

```
SQL> SELECT house, count(house) FROM houses GROUP BY house ORDER BY 1;

     HOUSE COUNT(HOUSE)
---------- ------------
         1            9
         2            5
         3            5

SQL> SELECT house, SKEWNESS_POP(price_big_city),
SKEWNESS_POP(price_small_city) FROM houses
        GROUP BY house;

     HOUSE SKEWNESS_POP(PRICE_BIG_CITY) SKEWNESS_POP(PRICE_SMALL_CITY)
---------- ---------------------------- ------------------------------
         1                            0                     -.66864012
         2                  1.13841996                     1.49637083
         3                            0                     -.12735442

SQL> SELECT house, SKEWNESS_SAMP(price_big_city),
SKEWNESS_SAMP(price_small_city) FROM houses
```

```
     GROUP BY house;

     HOUSE SKEWNESS_SAMP(PRICE_BIG_CITY) SKEWNESS_SAMP(PRICE_SMALL_CITY)
---------- ---------------------------- ------------------------------
         1                            0                      -.81051422
         2                   1.69705627                      2.23065793
         3                            0                      -.18984876

SQL>
```

Skewness is important in a situation where `PRICE_BIG_CITY` and `PRICE_SMALL_CITY` represent the prices of houses to buy and you want to determine whether the outliers in data are biased towards the left end or right end of the distribution, that is, if there are more values to the left of the mean when compared to the number of values to the right of the mean.

5. Insert more rows in the table.

```
SQL> INSERT INTO houses SELECT * FROM houses;

19 rows created.

SQL> /

38 rows created.

SQL> /

76 rows created.

SQL> /

152 rows created.

SQL> COMMIT;

Commit complete.

SQL> SELECT house, SKEWNESS_POP(price_big_city),
SKEWNESS_POP(price_small_city) FROM houses
     GROUP BY house ORDER BY 1;

     HOUSE SKEWNESS_POP(PRICE_BIG_CITY) SKEWNESS_POP(PRICE_SMALL_CITY)
---------- --------------------------- -----------------------------
         1                           0                     -.66864012
         2                  1.13841996                     1.49637083
         3                           0                     -.12735442

SQL> SELECT house, SKEWNESS_SAMP(price_big_city),
SKEWNESS_SAMP(price_small_city) FROM houses
     GROUP BY house ORDER BY 1;

     HOUSE SKEWNESS_SAMP(PRICE_BIG_CITY) SKEWNESS_SAMP(PRICE_SMALL_CITY)
---------- ---------------------------- ------------------------------
         1                            0                      -.67569912
```

```
         2                    1.1602897                    1.52511703
         3                            0                    -.12980098

SQL>
```

As you can see, as the number of values in the data set increases, the difference between the computed values of SKEWNESS_SAMP and SKEWNESS_POP decreases.

6. Determine the skewness of distinct values in columns PRICE_BIG_CITY and PRICE_SMALL_CITY.

```
SQL> SELECT house,
                    SKEWNESS_POP(DISTINCT price_big_city)
pop_big_city,
                    SKEWNESS_SAMP(DISTINCT price_big_city)
samp_big_city,
                    SKEWNESS_POP(DISTINCT price_small_city)
pop_small_city,
                    SKEWNESS_SAMP(DISTINCT price_small_city)
samp_small_city
              FROM houses
              GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1            0             0     -.66864012      -.81051422
         2   1.13841996    1.69705627     1.49637083      2.23065793
         3            0             0     -.12735442      -.18984876

SQL>
```

Is the result much different if the query does not evaluate the distinct values in columns PRICE_BIG_CITY and PRICE_SMALL_CITY?

```
SQL> SELECT house,
                    SKEWNESS_POP(price_big_city) pop_big_city,
                    SKEWNESS_SAMP(price_big_city) samp_big_city,
                    SKEWNESS_POP(price_small_city) pop_small_city,
                    SKEWNESS_SAMP(price_small_city) samp_small_city
              FROM houses
              GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1            0             0     -.66864012      -.67569912
         2   1.13841996     1.1602897     1.49637083      1.52511703
         3            0             0     -.12735442      -.12980098

SQL>
```

The population skewness value is not different because the same exact rows were inserted.

7. Insert more rows in the table with a big data set for HOUSE number 1.

```
SQL> INSERT INTO houses (house, price_big_city, price_small_city)
              SELECT house, price_big_city*0.5, price_small_city*0.1
              FROM houses WHERE house=1;

144 rows created.

SQL> /

288 rows created.

SQL> /

576 rows created.

SQL> /

1152 rows created.

SQL> /

2304 rows created.

SQL> COMMIT;

Commit complete.

SQL> SELECT house, count(house) FROM houses GROUP BY house ORDER BY 1;

      HOUSE COUNT(HOUSE)
---------- ------------
         1         4608
         2           80
         3           80

SQL> SELECT house,
                     SKEWNESS_POP(price_big_city) pop_big_city,
                     SKEWNESS_SAMP(price_big_city) samp_big_city,
                     SKEWNESS_POP(price_small_city) pop_small_city,
                     SKEWNESS_SAMP(price_small_city) samp_small_city
              FROM houses
              GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1   2.57050631    2.57134341      5.7418481      5.74371797
         2   1.13841996     1.1602897     1.49637083      1.52511703
         3            0             0     -.12735442      -.12980098

SQL>EXIT
$
```

Now the skewness becomes positive for house number 1 which means that data is skewed to right.

Practice: Measuring Tailedness of Data with the KURTOSIS Functions

This practice shows how to use the KURTOSIS_POP and KURTOSIS_SAMP aggregate functions to measure tailedness of data. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations. A normal distribution has a kurtosis of zero.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Connect to PDB20 as HR and execute the /home/oracle/labs/M104784GC10/Houses_Prices.sql SQL script. The SQL script creates a table with data.

```
$ cd /home/oracle/labs/M104784GC10
$ sqlplus hr@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 08:49:39 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Mon Mar 16 2020 08:48:58 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.
SQL> @/home/oracle/labs/M104784GC10/Houses_Prices.sql
SQL>
SQL> DROP TABLE houses;

Table dropped.

SQL> CREATE TABLE houses (house NUMBER, price_big_city NUMBER,
price_small_city NUMBER, price_date DATE);

Table created.

SQL> INSERT INTO houses VALUES (1,100000,10000, sysdate);

1 row created.

SQL> INSERT INTO houses VALUES (1,200000,15000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (1,300000,25000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (1,400000,28000, sysdate+2);

1 row created.
```

```
SQL> INSERT INTO houses VALUES (1,500000,30000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (1,600000,32000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (1,700000,35000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (1,800000,38000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (1,900000,40000, sysdate+5);

1 row created.

SQL>
SQL> INSERT INTO houses VALUES (2,2000000,1000000, sysdate+6);

1 row created.

SQL> INSERT INTO houses VALUES (2,200000,20000, sysdate);

1 row created.

SQL> INSERT INTO houses VALUES (2,400000,35000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (2,600000,55000, sysdate+1);

1 row created.

SQL> INSERT INTO houses VALUES (2,800000,48000, sysdate+2);

1 row created.

SQL>
SQL> INSERT INTO houses VALUES (3,400000,40000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (3,500000,42000, sysdate+3);

1 row created.

SQL> INSERT INTO houses VALUES (3,600000,45000, sysdate+4);

1 row created.
```

```
SQL> INSERT INTO houses VALUES (3,700000,48000, sysdate+4);

1 row created.

SQL> INSERT INTO houses VALUES (3,800000,49000, sysdate+5);

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

3. Display the table rows. The HOUSE column values refer to types of house that you want to look at and categorize the data that you look at statistically and compare with each other.

```
SQL> SET PAGES 100
SQL> SELECT * FROM houses;

     HOUSE PRICE_BIG_CITY PRICE_SMALL_CITY PRICE_DAT
---------- -------------- ---------------- ---------
         1         100000            10000 06-FEB-20
         1         200000            15000 07-FEB-20
         1         300000            25000 07-FEB-20
         1         400000            28000 08-FEB-20
         1         500000            30000 09-FEB-20
         1         600000            32000 09-FEB-20
         1         700000            35000 10-FEB-20
         1         800000            38000 10-FEB-20
         1         900000            40000 11-FEB-20
         2        2000000          1000000 12-FEB-20
         2         200000            20000 06-FEB-20
         2         400000            35000 07-FEB-20
         2         600000            55000 07-FEB-20
         2         800000            48000 08-FEB-20
         3         400000            40000 09-FEB-20
         3         500000            42000 09-FEB-20
         3         600000            45000 10-FEB-20
         3         700000            48000 10-FEB-20
         3         800000            49000 11-FEB-20

19 rows selected.

SQL>
```

4. Display the result of population kurtosis (KURTOSIS_POP) and sample kurtosis (KURTOSIS_SAMP) for the three types of houses.

```
SQL> SELECT house, kurtosis_pop(price_big_city),
kurtosis_pop(price_small_city) FROM houses
      GROUP BY house;

     HOUSE KURTOSIS_POP(PRICE_BIG_CITY) KURTOSIS_POP(PRICE_SMALL_CITY)
```

```
---------- -------------------------- ----------------------------
         1                      -1.23                     -.7058169
         2                      -.212                    .245200191
         3                       -1.3                    -1.5417881

SQL> SELECT house, kurtosis_samp(price_big_city),
kurtosis_samp(price_small_city) FROM houses
     GROUP BY house;

     HOUSE KURTOSIS_SAMP(PRICE_BIG_CITY) KURTOSIS_SAMP(PRICE_SMALL_CITY)
---------- -------------------------- ----------------------------
         1                       -1.2                      -.201556
         2                      3.152                    4.98080076
         3                       -1.2                    -2.1671526

SQL>
```

PRICE_SMALL_CITY has a higher kurtosis compared to PRICE_BIG_CITY. Observe whether there is more data in the tails or around the peak in PRICE_SMALL_CITY and in PRICE_BIG_CITY.

5. Insert more rows in the table.

```
SQL> INSERT INTO houses SELECT * FROM houses;

19 rows created.

SQL> /

38 rows created.

SQL> /

76 rows created.

SQL> /

152 rows created.

SQL> COMMIT;

Commit complete.

SQL> SELECT house, KURTOSIS_POP(price_big_city),
KURTOSIS_POP(price_small_city) FROM houses
     GROUP BY house ORDER BY 1;

     HOUSE KURTOSIS_POP(PRICE_BIG_CITY) KURTOSIS_POP(PRICE_SMALL_CITY)
---------- -------------------------- ----------------------------
         1                      -1.23                     -.7058169
         2                      -.212                    .245200191
         3                       -1.3                    -1.5417881

SQL> SELECT house, KURTOSIS_SAMP(price_big_city),
KURTOSIS_SAMP(price_small_city) FROM houses
```

```
     GROUP BY house ORDER BY 1;

     HOUSE KURTOSIS_SAMP(PRICE_BIG_CITY) KURTOSIS_SAMP(PRICE_SMALL_CITY)
---------- ---------------------------- ------------------------------
         1                   -1.2309485                     -.68809876
         2                   -.14695105                      .340165838
         3                   -1.3061439                     -1.5637533

SQL>
```

As you can see, as the number of values in the data set increases, the difference between the computed values of KURTOSIS_SAMP and KURTOSIS_POP decreases.

6. Determine the kurtosis of distinct values in columns PRICE_SMALL_CITY and PRICE_BIG_CITY.

```
SQL> SELECT house,
                    KURTOSIS_POP(DISTINCT price_big_city)
pop_big_city,
                    KURTOSIS_SAMP(DISTINCT price_big_city)
samp_big_city,
                    KURTOSIS_POP(DISTINCT price_small_city)
pop_small_city,
                    KURTOSIS_SAMP(DISTINCT price_small_city)
samp_small_city
              FROM houses
              GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1        -1.23          -1.2      -.7058169        -.201556
         2        -.212         3.152     .245200191      4.98080076
         3         -1.3          -1.2     -1.5417881      -2.1671526

SQL>
```

Is the result much different if the query does not evaluate the distinct values in columns PRICE_BIG_CITY and PRICE_SMALL_CITY?

```
SQL> SELECT house,
                    KURTOSIS_POP(price_big_city) pop_big_city,
                    KURTOSIS_SAMP(price_big_city) samp_big_city,
                    KURTOSIS_POP(price_small_city) pop_small_city,
                    KURTOSIS_SAMP(price_small_city) samp_small_city
              FROM houses
              GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1        -1.23    -1.2309485      -.7058169      -.68809876
         2        -.212    -.14695105     .245200191      .340165838
         3         -1.3    -1.3061439     -1.5417881      -1.5637533

SQL>
```

The population tailedness value is not different because the same exact rows were inserted.

7. Insert more rows in the table with a big data set for HOUSE number 1.

```
SQL> INSERT INTO houses (house, price_big_city, price_small_city)
             SELECT house, price_big_city*0.5, price_small_city*0.1
             FROM houses WHERE house=1;

144 rows created.

SQL> /

288 rows created.

SQL> /

576 rows created.

SQL> /

1152 rows created.

SQL> /

2304 rows created.

SQL> COMMIT;

Commit complete.

SQL> SELECT house, count(house) FROM houses GROUP BY house ORDER BY 1;

      HOUSE COUNT(HOUSE)
---------- ------------
         1         4608
         2           80
         3           80

SQL> SELECT house,
                    KURTOSIS_POP(price_big_city) pop_big_city,
                    KURTOSIS_SAMP(price_big_city) samp_big_city,
                    KURTOSIS_POP(price_small_city) pop_small_city,
                    KURTOSIS_SAMP(price_small_city) samp_small_city
             FROM houses
             GROUP BY house;

     HOUSE POP_BIG_CITY SAMP_BIG_CITY POP_SMALL_CITY SAMP_SMALL_CITY
---------- ------------ ------------- -------------- ---------------
         1   9.12746931    9.13868421     33.7452495      33.7831972
         2         -.212    -.14695105     .245200191      .340165838
         3         -1.3    -1.3061439     -1.5417881      -1.5637533

SQL>EXIT
$
```

Now the tailedness of the data becomes positive for house number 1 which means that data is skewed to right.

PRICE_SMALL_CITY has a much higher kurtosis compared to PRICE_BIG_CITY. This implies that in PRICE_SMALL_CITY, more of the variance is the result of many infrequent extreme deviations, whereas in PRICE_BIG_CITY, the variance is attributed to very frequent modestly sized deviations.

Enhanced Analytic Functions

Window functions now support the EXCLUDE options of the SQL standard window frame clause. The query_block clause of a SELECT statement now supports the window_clause, which implements the window clause of the SQL standard table expression as defined in the SQL:2011 standard.

Supporting the full ANSI standard enables easier migration of applications that were developed against other standard-compliant database systems.

- Practice: Using Enhanced Analytic Functions
  This practice shows how to benefit from the new options of the window frame clause, GROUPS and EXCLUDE, and also from the WINDOW clause in the table expression.

Related Topics

- *Oracle® Database Data Warehousing Guide*

Practice: Using Enhanced Analytic Functions

This practice shows how to benefit from the new options of the window frame clause, GROUPS and EXCLUDE, and also from the WINDOW clause in the table expression.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Experiment the usage of the GROUPS clause of the window frame.
   a. Before starting experimenting the usage of the GROUPS clause of the window frame, execute the /home/oracle/labs/M104784GC10/setup_analytic_table.sh shell script. The shell script creates in both PDB20 and PDB19 the user REPORT, grants REPORT the CREATE SESSION, CREATE TABLE and UNLIMITED TABLESPACE privileges, and finally creates the table TRADES including rows.

```
$ /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/setup_analytic_table.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Feb 3 09:23:40 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> DROP USER report CASCADE;
```

```
User dropped.

SQL> CREATE USER report IDENTIFIED BY password;

User created.

SQL> GRANT create session, create table, unlimited tablespace TO
report;

Grant succeeded.

SQL> CREATE TABLE report.trades (acno NUMBER, tid NUMBER, Tday
DATE, Ttype VARCHAR2(4), amount NUMBER, Ticker VARCHAR2(4));

Table created.

SQL> INSERT INTO report.trades VALUES (123, 1, sysdate, 'buy',
1000, 'CSCO');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 1, sysdate, 'buy', 400,
'JNPR');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 3, sysdate+2, 'buy',
2000, 'SYMC');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 4, sysdate+2, 'buy',
1200, 'CSCO');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 5, sysdate+2, 'buy',
500, 'JNPR');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 6, sysdate+4, 'buy',
200, 'CSCO');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 7, sysdate+4, 'buy',
100, 'CSCO');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 9, sysdate+5, 'buy',
400, 'JNPR');

1 row created.
```

```
SQL> INSERT INTO report.trades VALUES (123, 10, sysdate+5, 'buy',
200, 'GOOG');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 11, sysdate+5, 'buy',
1000, 'JNPR');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 12, sysdate+5, 'buy',
4000, 'JNPR');

1 row created.

SQL> INSERT INTO report.trades VALUES (123, 13, sysdate+8, 'buy',
2000, 'HPQ');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

b. Display the rows of `REPORT.TRADES` in `PDB20`. Using `ROWS`, the user specifies the window frame extent by counting rows forward or backward from the current row. `ROWS` allows any number of sort keys, of any ordered data types. This can be advantageous, because counting rows is oblivious to any “holes” in the values that are sorted. On the other hand, counting rows from the current row can be non-deterministic when there are multiple rows that are identical in the sort keys, causing an arbitrary cutoff between two rows that have the same values in the sort keys. Using `RANGE`, the user specifies an offset. There must be precisely one sort key, and its declared type must be amenable to addition and subtraction (i.e., numeric,datetime or interval). This avoids the non-determinism of arbitrarily cutting between two adjacent rows with the same value, but it can only be used with a single sort key of an additive type. SQL:2011 standard includes a third way of specifying the window frame extent, using the keyword `GROUPS`. Like `ROWS`, a `GROUPS` window can have any number of sort keys, of any ordered types. Like `RANGE`, a `GROUPS` window does not make cutoffs between adjacent rows with the same values in the sort keys. Thus, `GROUPS` combines some of the features of both `ROWS` and `RANGE`.

```
$ sqlplus report@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Feb 3 09:31:17 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.
```

```
Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0
SQL> SET PAGES 100
SQL> SELECT * FROM trades;

      ACNO        TID TDAY      TTYP     AMOUNT TICK
---------- ---------- --------- ---- ---------- ----
       123          1 08-APR-20 buy        1000 CSCO
       123          1 08-APR-20 buy         400 JNPR
       123          3 10-APR-20 buy        2000 SYMC
       123          4 10-APR-20 buy        1200 CSCO
       123          5 10-APR-20 buy         500 JNPR
       123          6 12-APR-20 buy         200 CSCO
       123          7 12-APR-20 buy         100 CSCO
       123          9 13-APR-20 buy         400 JNPR
       123         10 13-APR-20 buy         200 GOOG
       123         11 13-APR-20 buy        1000 JNPR
       123         12 13-APR-20 buy        4000 JNPR
       123         13 16-APR-20 buy        2000 HPQ

12 rows selected.

SQL>
```

c. Compute the total amount over the last five days on which account number 123 performed a “buy”. To answer this query, you can group the data by trade day, compute the sum of amount on each trade day, and then use a ROWS window to add up the last five trade days.

```
SQL> SELECT trades.acno, trades.tday, SUM (agg.suma) OVER W
FROM    trades, (SELECT acno, tday, SUM(amount) AS suma
                 FROM   trades
              WHERE  ttype = 'buy'
              GROUP BY acno, tday ) agg
WHERE   trades.acno = agg.acno
AND     trades.tday = agg.tday
AND     trades.ttype = 'buy'
WINDOW W AS (PARTITION BY trades.acno ORDER BY trades.tday ROWS
BETWEEN 4 PRECEDING AND CURRENT ROW);

      ACNO TDAY      SUM(AGG.SUMA)OVERW
---------- --------- ------------------
       123 08-APR-20               1400
       123 08-APR-20               2800
       123 10-APR-20               6500
       123 10-APR-20              10200
       123 10-APR-20              13900
       123 12-APR-20              12800
       123 12-APR-20              11700
       123 13-APR-20              13600
```

```
       123 13-APR-20             15500
       123 13-APR-20             17400
       123 13-APR-20             22700
       123 16-APR-20             24400

12 rows selected.

SQL>
```

The reason why this query works is because it is possible to decompose a sum into partial aggregates, and compute the final sum from those partial aggregates. In this case, the query is decomposing the sum over groups defined by `acno` and `tday`. Then the query gets the sum over 5 trading days by adding the partial sums from the grouped query. `COUNT`, `MAX` and `MIN` are also decomposable aggregates. `AVG` can be decomposed by computing sums and counts and then dividing.

When the window name is specified with a windowing clause, it can only be referenced directly, without parentheses.

d. Query how many distinct ticker symbols were traded in the preceding 5 trading days. This requires a `COUNT DISTINCT`, which cannot be decomposed into partial counts, one for each trading day, because there may be duplicate ticker symbols on different trading days, as can be seen in the sample data. `COUNT DISTINCT` is not decomposable, and the technique in the preceding query cannot be used. Use the keyword `GROUPS` instead of `RANGE` or `ROWS`. The keyword `GROUPS` emphasizes the relationship to grouped queries. Using this kind of keyword, we can answer queries such as, for each account number, for the last five trading days on which the account executed a “buy”, find the amount spent and the number of distinct ticker symbols bought.

```
SQL> SELECT acno, tday, SUM(amount) OVER W, COUNT(DISTINCT ticker)
OVER W
FROM   trades
WHERE  ttype = 'buy'
WINDOW W AS (PARTITION BY acno ORDER BY tday GROUPS BETWEEN 4
PRECEDING AND CURRENT ROW);
SELECT acno, tday, SUM(amount) OVER W, COUNT(DISTINCT ticker) OVER W
                                                               *
ERROR at line 1:
ORA-30487: ORDER BY not allowed here


SQL>
```

> **Note:**
>
> <*aggregate function*> with `DISTINCT` specification cannot be used with <*window specification*> having <*window order clause*>.

```
SQL> SELECT acno, tday, SUM(amount) OVER W, COUNT(ticker) OVER W
FROM   trades
```

```
WHERE  ttype = 'buy'
WINDOW W AS (PARTITION BY acno ORDER BY tday GROUPS BETWEEN 4
PRECEDING AND CURRENT ROW);

      ACNO TDAY      SUM(AMOUNT)OVERW COUNT(TICKER)OVERW
---------- --------- ---------------- ------------------
       123 08-APR-20             1400                  2
       123 08-APR-20             1400                  2
       123 10-APR-20             5100                  5
       123 10-APR-20             5100                  5
       123 10-APR-20             5100                  5
       123 12-APR-20             5400                  7
       123 12-APR-20             5400                  7
       123 13-APR-20            11000                 11
       123 13-APR-20            11000                 11
       123 13-APR-20            11000                 11
       123 13-APR-20            11000                 11
       123 16-APR-20            13000                 12

12 rows selected.

SQL>
```

Notice that the syntax avoids the need for a nested grouped query and a join with TRADES as it was the case in step c.

3. Experiment the usage of the EXCLUDE clause of the window frame.
   a. Before starting experimenting the usage of the EXCLUDE clause of the window frame, execute the /home/oracle/labs/M104784GC10/create_T_table.sql SQL script.

```
SQL> @/home/oracle/labs/M104784GC10/create_T_table.sql
SQL> SET ECHO ON
SQL> DROP TABLE t;

Table dropped.

SQL> CREATE TABLE t (v NUMBER);

Table created.

SQL> INSERT INTO t VALUES (1);

1 row created.

SQL> INSERT INTO t VALUES (1);

1 row created.

SQL> INSERT INTO t VALUES (3);

1 row created.

SQL> INSERT INTO t VALUES (5);
```

```
1 row created.

SQL> INSERT INTO t VALUES (5);

1 row created.

SQL> INSERT INTO t VALUES (5);

1 row created.

SQL> INSERT INTO t VALUES (6);

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

b. Display the rows of table T.

```
SQL> SELECT * FROM t;

         V
----------
         1
         1
         3
         5
         5
         5
         6

7 rows selected.

SQL>
```

c. Use the EXCLUDE options for window frame exclusion with ROWS. If EXCLUDE CURRENT ROW is specified and the current row is still a member of the window frame, then remove the current row from the window frame. If EXCLUDE GROUP is specified, then remove the current row and any peers of the current row from the window frame. If EXCLUDE TIES is specified, then remove any rows other than the current row that are peers of the current row from the window frame. If the current row is already removed from the window frame, then it remains removed from the window frame. If EXCLUDE NO OTHERS is specified (this is the default), then no additional rows are removed from the window frame by this rule.

```
SQL> SELECT v,
                sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE CURRENT ROW) AS current_row,
                sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE GROUP) AS the_group,
                sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1
```

```
FOLLOWING EXCLUDE TIES) AS ties,
               sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE NO OTHERS) AS no_others
         FROM t
         WINDOW o AS (ORDER BY v);

         V CURRENT_ROW  THE_GROUP       TIES  NO_OTHERS
---------- ----------- ---------- ---------- ----------
         1           1                     1          2
         1           4          3          4          5
         3           6          6          9          9
         5           8          3          8         13
         5          10                     5         15
         5          11          6         11         16
         6           5          5         11         11

7 rows selected.

SQL> SELECT v,
               sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2
FOLLOWING EXCLUDE CURRENT ROW) AS current_row,
               sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2
FOLLOWING EXCLUDE GROUP) AS the_group,
               sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2
FOLLOWING EXCLUDE TIES) AS ties,
               sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2
FOLLOWING EXCLUDE NO OTHERS) AS no_others
         FROM t
         WINDOW o AS (ORDER BY v);

         V CURRENT_ROW  THE_GROUP       TIES  NO_OTHERS
---------- ----------- ---------- ---------- ----------
         1           4          3          4          5
         1           9          8          9         10
         3          12         12         15         15
         5          14          4          9         19
         5          19          9         14         24
         5          16          6         11         21
         6          10         10         16         16

7 rows selected.

SQL>
```

d. Use the EXCLUDE options for window frame exclusion with RANGE.

```
SQL> SELECT v,
               sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE CURRENT ROW) AS current_row,
               sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE GROUP) AS the_group,
               sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE TIES) AS ties,
               sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE NO OTHERS) AS no_others
```

```
        FROM t
        WINDOW o AS (ORDER BY v);

        V CURRENT_ROW  THE_GROUP       TIES  NO_OTHERS
---------- ----------- ---------- ---------- ----------
        1           1                     1          2
        1           1                     1          2
        3                                 3          3
        5          16          6         11         21
        5          16          6         11         21
        5          16          6         11         21
        6          15         15         21         21

7 rows selected.

SQL>
```

e. Use the `EXCLUDE` options for window frame exclusion with `GROUPS`.

```
SQL> SELECT v,
                sum(v) OVER (o GROUPS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE CURRENT ROW) AS current_row,
                sum(v) OVER (o GROUPS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE GROUP) AS the_group,
                sum(v) OVER (o GROUPS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE TIES) AS ties,
                sum(v) OVER (o GROUPS BETWEEN 1 PRECEDING AND 1
FOLLOWING EXCLUDE NO OTHERS) AS no_others
        FROM t
        WINDOW o AS (ORDER BY v);

        V CURRENT_ROW  THE_GROUP       TIES  NO_OTHERS
---------- ----------- ---------- ---------- ----------
        1           4          3          4          5
        1           4          3          4          5
        3          17         17         20         20
        5          19          9         14         24
        5          19          9         14         24
        5          19          9         14         24
        6          15         15         21         21

7 rows selected.

SQL> EXIT
$
```

SQL

- SQL Macros
- Placeholders in SQL DDL Statements
- Expression Support for Initialization Parameters
- Enhanced SQL Set Operators

SQL Macros

You can create SQL Macros (SQM) to factor out common SQL expressions and statements into reusable, parameterized constructs that can be used in other SQL statements. SQL macros can either be scalar expressions, typically used in `SELECT` lists, `WHERE`, `GROUP BY` and `HAVING` clauses, to encapsulate calculations and business logic or can be table expressions, typically used in a `FROM` clause.

SQL macros increase developer productivity, simplify collaborative development, and improve code quality.

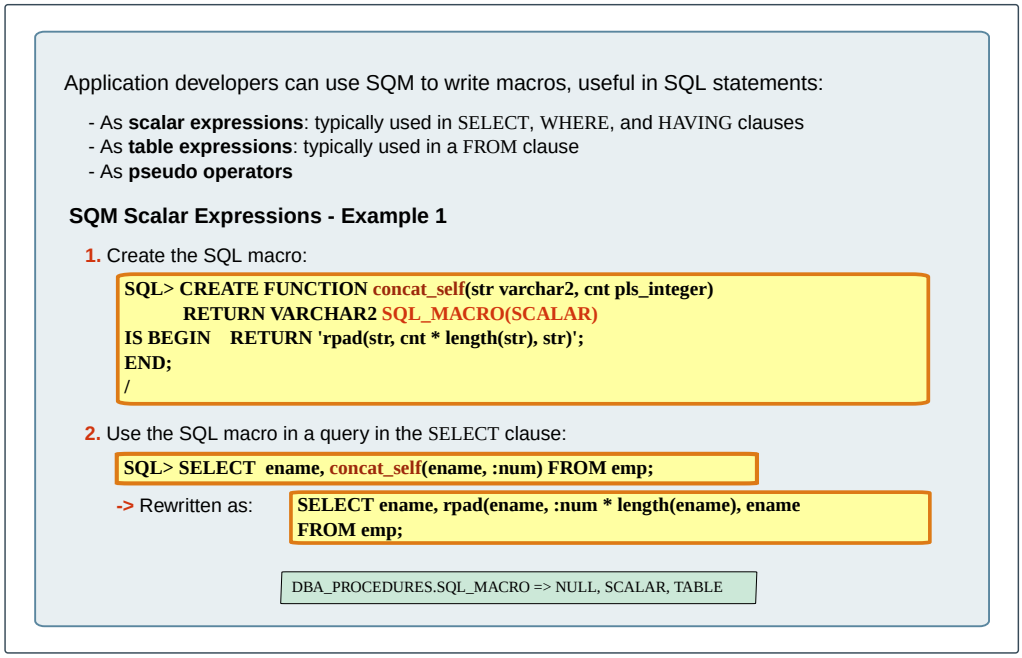
- Details: SQL Macros
  This page provides more detailed information explaining what SQL Macros are useful for and supplying various examples to use.
- Practice: Using SQM Scalar and Table Expressions
  These practices show how to use SQL Macro as scalar and table expressions.

Related Topics

- *Oracle® Database PL/SQL Language Reference*

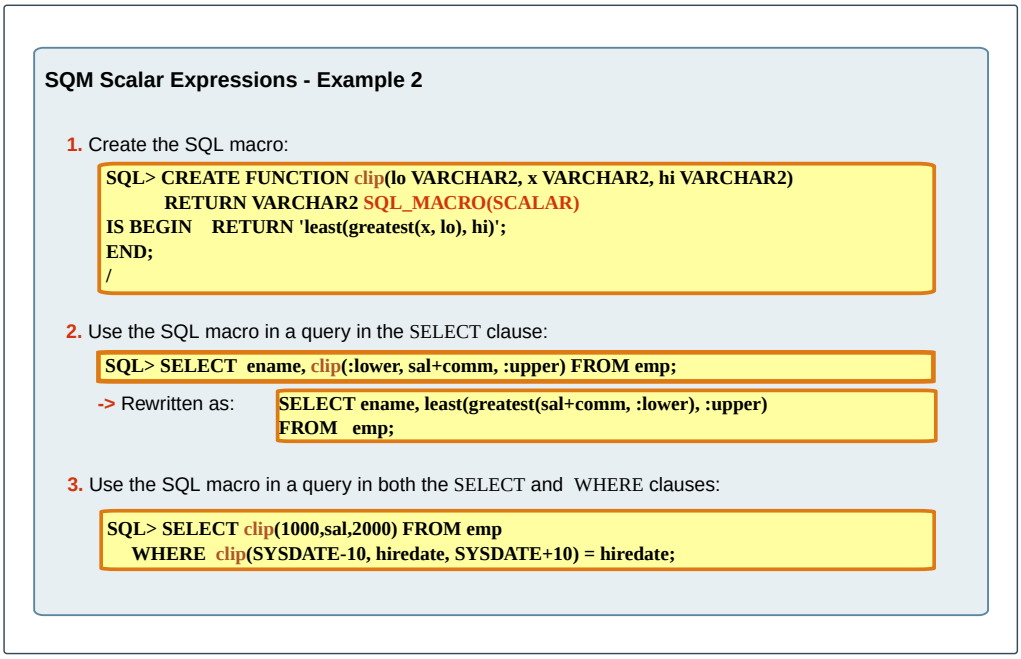
Details: SQL Macros

This page provides more detailed information explaining what SQL Macros are useful for and supplying various examples to use.

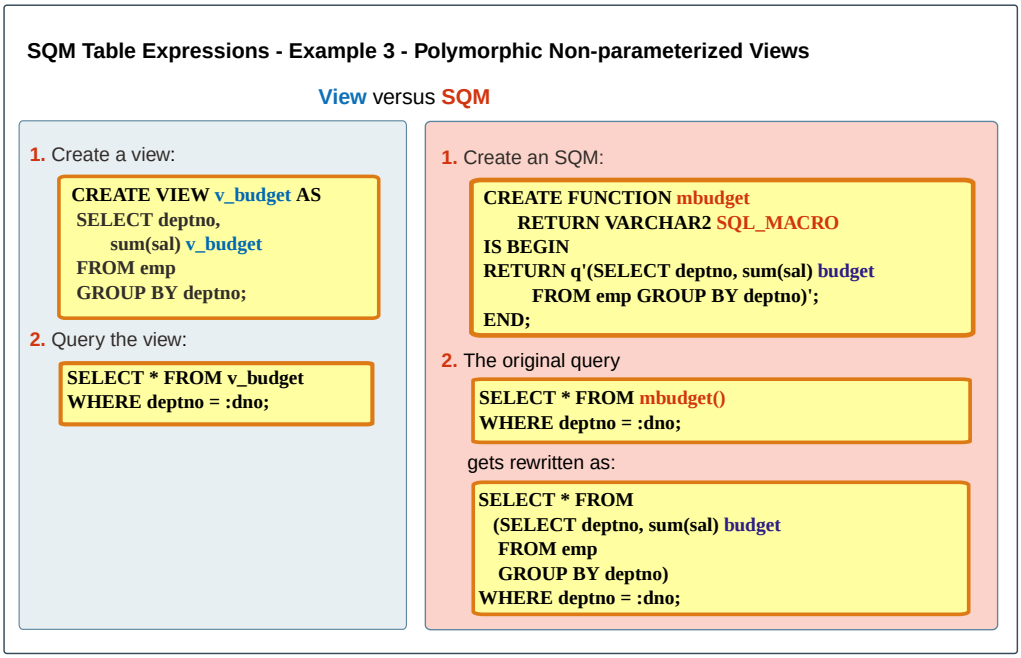


You can create SQL macros (SQM) to factor out common SQL expressions and statements into reusable, parametized constructs that can be used in other SQL statements. SQL macros can either be scalar expressions, typically used in `SELECT` lists, `WHERE`, and `HAVING` clauses, to encapsulate calculations and business logic, or can be table expressions, typically used in a `FROM` clause, to act as a sort of parametized views. SQL macros increase developer productivity, simplifies collaborative development, and improves code quality.

The example in the slide shows an SQL macro written as a scalar expression, used in the SELECT list of the query.



The example in the slide shows an SQL macro written as a scalar expression, used in the SELECT list and the WHERE clause of the query.



The example in the slide shows an SQL macro written as a table expression, then used in the FROM list of the query.

Practice: Using SQM Scalar and Table Expressions

These practices show how to use SQL Macro as scalar and table expressions.

1. Before starting any new practice, refer to the practices environment recommendations.
2. The first practice is an easy one to show you how to concatenate an employee name to its own name as many times as defined during the execution of the SQL macro.
   a. Create the HR schema and its tables.

```
$ sqlplus sys@pdb20 AS SYSDBA
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Apr 1 12:32:01 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> @$ORACLE_HOME/demo/schema/human_resources/hr_main.sql password
users temp /home/oracle/labs /home/oracle/labs
specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:

PL/SQL procedure successfully completed.

User created.

User altered.

User altered.

Grant succeeded.

Grant succeeded.
...

Commit complete.

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

   b. Create the SQM as an scalar expression.

```
$ sqlplus hr@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 10:37:50
2020
```

```
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Mon Mar 16 2020 09:27:07 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> CREATE OR REPLACE FUNCTION concat_self(str varchar2, cnt
pls_integer)
            RETURN VARCHAR2 SQL_MACRO(SCALAR)
      IS BEGIN
            RETURN 'rpad(str, cnt * length(str), str)';
END;
/
Function created.

SQL>
```

c. Use the SQM to query the table and display the employees names doubled.

```
SQL> COL CONCAT_SELF(LAST_NAME,2) FORMAT A40
SQL> SELECT last_name, concat_self(last_name,2) FROM hr.employees;

LAST_NAME                 CONCAT_SELF(LAST_NAME,2)
------------------------- ----------------------------------------
Abel                      AbelAbel
Ande                      AndeAnde
Atkinson                  AtkinsonAtkinson
Austin                    AustinAustin
Baer                      BaerBaer
Baida                     BaidaBaida
Banda                     BandaBanda
Bates                     BatesBates
Bell                      BellBell
Bernstein                 BernsteinBernstein
Bissot                    BissotBissot
...
107 rows selected.

SQL>
```

d. Use the SQM to query the table and display the employees names tripled.

```
SQL> COL CONCAT_SELF(LAST_NAME,3) FORMAT A40
SQL> SELECT last_name, concat_self(last_name,3) FROM hr.employees;

LAST_NAME                 CONCAT_SELF(LAST_NAME,3)
------------------------- ----------------------------------------
Abel                      AbelAbelAbel
Ande                      AndeAndeAnde
```

```
Atkinson                  AtkinsonAtkinsonAtkinson
Austin                    AustinAustinAustin
Baer                      BaerBaerBaer
Baida                     BaidaBaidaBaida
Banda                     BandaBandaBanda
Bates                     BatesBatesBates
Bell                      BellBellBell
Bernstein                 BernsteinBernsteinBernstein
Bissot                    BissotBissotBissot
Bloom                     BloomBloomBloom
Bull                      BullBullBull
Cabrio                    CabrioCabrioCabrio
...
107 rows selected.

SQL>
```

3. The second practice shows how to use an SQM as a table expression to implement a polymorphic view.

   a. Let us use a simple view to display the sum of the salaries per department.

```
SQL> CREATE VIEW v_budget
 AS SELECT department_id, sum(salary) v_budget
    FROM hr.employees
    GROUP BY department_id;

View created.

SQL>
```

   b. Query the result from the view.

```
SQL> SELECT * FROM v_budget WHERE department_id IN (10,50);

DEPARTMENT_ID   V_BUDGET
------------- ----------
           50     156400
           10       4400

SQL>
```

   c. Now use an SQM as a table expression. Create the SQM.

```
SQL> CREATE OR REPLACE FUNCTION budget
return varchar2 SQL_MACRO
IS
BEGIN
  RETURN q'( select department_id, sum(salary) budget
             from hr.employees
             group by department_id )';
END;
/

Function created.
```

```
SQL>
```

d. Use the SQM to display the result for the departments 10 and 50.

```
SQL> SELECT * FROM budget() WHERE department_id IN (10,50);

DEPARTMENT_ID     BUDGET
------------- ----------
           50     156400
           10       4400

SQL>
```

4. The third practice shows how to use an SQM as a table expression to display sum of the salaries per department for a particular job.

   a. Create the SQM.

```
SQL> CREATE OR REPLACE FUNCTION budget_per_job(job_id varchar2)
return varchar2 SQL_MACRO
IS
BEGIN
  RETURN q'( select department_id, sum(salary) budget
             from hr.employees
             where job_id = budget_per_job.job_id
             group by department_id )';
END;
/
Function created.

SQL>
```

   b. Use the SQM to display the result for the `ST_CLERK` job in department 10.

```
SQL> SELECT * FROM budget_per_job('ST_CLERK') WHERE department_id =
10;

no rows selected

SQL>
```

   c. Use the SQM to display the result for the `SH_CLERK` job in department 50.

```
SQL> SELECT * FROM budget_per_job('SH_CLERK') WHERE department_id =
50;

DEPARTMENT_ID BUDGET_PER_JOB
------------- --------------
           50          64300

SQL>
```

d. Use the DBMS_OUTPUT package to display the rewritten SQL query. Recreate the function including the DBMS_OUTPUT package.

```
SQL> CREATE OR REPLACE function budget_per_job(job_id varchar2)
return varchar2 SQL_MACRO
is
  stmt varchar(2000) := q'(
   select department_id, sum(salary) budget
   from hr.employees
   where job_id = budget_per_job.job_id
   group by department_id )';
begin

dbms_output.put_line('-----------------------------------------
');
  dbms_output.put_line('SQM Text: ' );

dbms_output.put_line('-----------------------------------------
');
  dbms_output.put_line('  ' ||stmt);

dbms_output.put_line('-----------------------------------------
');
  return stmt;
end;
/
Function created.

SQL>
```

e. Re-execute the query using the SQM.

```
SQL> SET serveroutput on
SQL> SET LONG 20000
SQL> SELECT * FROM budget_per_job('ST_CLERK') WHERE department_id =
50;

DEPARTMENT_ID     BUDGET
------------- ----------
           50      55700

-----------------------------------------
SQM Text:
-----------------------------------------

   select department_id, sum(salary) budget
   from hr.employees
   where
job_id = budget_per_job.job_id
   group by department_id
-----------------------------------------
SQL>
```

5. Use the USER_PROCEDURES view to display the new values of the SQL_MACRO column.

```
SQL> COL object_name FORMAT A30
SQL> SELECT object_name, sql_macro, object_type FROM user_procedures;

OBJECT_NAME                    SQL_MA OBJECT_TYPE
------------------------------ ------ -------------
SECURE_DML                     NULL   PROCEDURE
BUDGET                         TABLE  FUNCTION
ADD_JOB_HISTORY                NULL   PROCEDURE
BUDGET_PER_JOB                 TABLE  FUNCTION
CONCAT_SELF                    SCALAR FUNCTION
SECURE_EMPLOYEES                      TRIGGER
UPDATE_JOB_HISTORY                    TRIGGER

7 rows selected.

SQL> EXIT
$
```

Placeholders in SQL DDL Statements

SQL DDL statements can now contain placeholders instead of hard coded values for some content. For example, placeholders may be used where a username or password are required in a CREATE USER statement. Oracle Call Interface programs can substitute values into the DDL statement placeholders before the statements are sent to Oracle Database. This is similar to data binding, but occurs in Oracle Client.

Application security is improved because values do not need to be hard coded in SQL DDL.

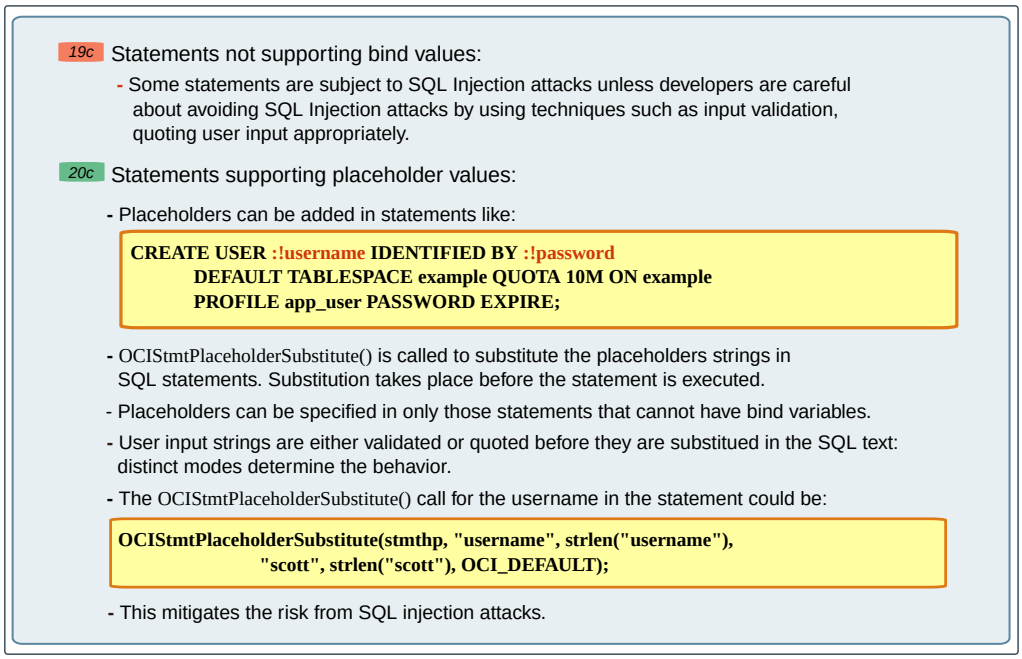
- Details: Placeholders in SQL Statements
  This page provides more detailed information about the `OCIStmtPlaceholderSubstitute()` function. The `OCIStmtPlaceholderSubstitute()` substitutes placeholder strings in SQL statements. Placeholders can be specified in only those statements that cannot have bind variables. OCI placeholders are not the same as bind variables.

Related Topics

- *Oracle® Call Interface Programmer's Guide*

Details: Placeholders in SQL Statements

This page provides more detailed information about the `OCIStmtPlaceholderSubstitute()` function. The `OCIStmtPlaceholderSubstitute()` substitutes placeholder strings in SQL statements. Placeholders can be specified in only those statements that cannot have bind variables. OCI placeholders are not the same as bind variables.



The statements that cannot have OCI placeholders are those beginning with the keywords like `SELECT`, `UPDATE`, `DELETE`, `INSERT`, `BEGIN`, `DECLARE`, `RETURNING`, `CALL`, `MERGE`, `ROLLBACK`, `COMMIT`, and `FLASHBACK` since they support bind variables. Other SQL statements such as `CREATE`, `DROP`, `ALTER`, `EXPLAIN` statements can have OCI placeholders.

The parameters of the `OCIStmtPlaceholderSubstitute()` function are defined in the Oracle® Call Interface Programmer's Guide 20c.

Expression Support for Initialization Parameters

You can specify an expression when setting the value of an initialization parameter.

In previous releases, you were required to specify an absolute value when setting an initialization parameter. You can now specify an expression that takes into account the current system configuration and environment. This is especially useful in Oracle Autonomous Database environments.

- Practice: Using Expressions in Initialization Parameters
  This practice shows how to optimize the values set in initialization parameters when they depend on environmental characteristics, such as system configurations, run-time decisions, or the values of other parameters by using expressions.

Related Topics

- *Oracle® Database Reference*

Practice: Using Expressions in Initialization Parameters

This practice shows how to optimize the values set in initialization parameters when they depend on environmental characteristics, such as system configurations, run-time decisions, or the values of other parameters by using expressions.

1. Before starting any new practice, refer to the practices environment recommendations.

2. Log in to PDB20 as SYSTEM.

```
$ sqlplus system
SQL*Plus: Release 20.0.0.0.0 - Production on Thu Jan 9 04:08:41 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Wed Jan 08 2020 12:03:56 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL>
SQL>
```

3. Set the sga_target to 2G.

```
SQL> ALTER SYSTEM SET sga_target = 2G;
ALTER SYSTEM SET sga_target = 2G
*
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value is
invalid
ORA-00823: Specified value of sga_target greater than sga_max_size

SQL>
```

As it fails, set it to 80 % of the SGA_MAX_SIZE.

```
SQL> ALTER SYSTEM SET sga_target = 'sga_max_size*80/100';

System altered.

SQL> SHOW PARAMETER sga

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
allow_group_access_to_sga            boolean     FALSE
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 1360M
sga_min_size                         big integer 0
sga_target                           big integer 1088M
SQL>
```

4. Set the job_queue_processes to the 10% of the processes value.

```
SQL> ALTER SYSTEM SET job_queue_processes='processes*10/100' SCOPE=BOTH;

System altered.
```

```
SQL> SHOW PARAMETER processes

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
aq_tm_processes                      integer     1
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     32
log_archive_max_processes            integer     4
processes                            integer     320
SQL>
```

5. Set the aq_tm_processes to the minimum value between 40 and 10% of processes.

```
SQL> ALTER SYSTEM SET AQ_TM_PROCESSES = 'MIN(40, PROCESSES * .1)'
SCOPE=BOTH;

System altered.

SQL> SHOW PARAMETER processes

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
aq_tm_processes                      integer     32
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     320
log_archive_max_processes            integer     4
processes                            integer     320

SQL>
```

6. What happens if you change the value of processes?
   a. Set the processes value to 500 in SPFILE.

```
SQL> ALTER SYSTEM SET PROCESSES = 500 SCOPE=SPFILE;

System altered.

SQL>
```

   b. Restart the CDB instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
```

```
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area 1140848912 bytes
Fixed Size                  9566480 bytes
Variable Size             352321536 bytes
Database Buffers          771751936 bytes
Redo Buffers                7208960 bytes
Database mounted.
Database opened.
SQL>
```

c. Display the values for `processes` and `aq_tm_processes`.

```
SQL> SHOW PARAMETER processes

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
aq_tm_processes                      integer     40
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     50
log_archive_max_processes            integer     4
processes                            integer     500
SQL>
```

The minimum value between 40 and 10% of `processes` is now 40 (because 10% of 500 is 50). The expression used for setting the `aq_tm_processes` parameter is kept throughout the database instance restarts.

7. Set the `db_recovery_file_dest` to the same value as `$HOME`, in `CDB20`.

```
SQL> ALTER SYSTEM SET db_recovery_file_dest='$HOME' SCOPE=BOTH;

System altered.

SQL> SHOW PARAMETER db_recovery_file_dest

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_recovery_file_dest                string      $HOME
db_recovery_file_dest_size           big integer 15000M
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;
```

```
SQL> HOST
$ cd $HOME
$ ls -ltR | more
.:
total 20
drwxr-x--- 3 oracle oinstall 4096 Apr  8 11:49 CDB20_IAD3CV
drwxrwxrwx 9 oracle oinstall 4096 Apr  8 10:11 labs
drwxrwxrwx 2 oracle oinstall 4096 Apr  3 13:06 foo
-rwxrwxrwx 1 oracle oinstall  590 Apr  3 10:27 database2007112852029
274968.rsp
-rwxrwxrwx 1 oracle oinstall  668 Apr  3 10:27 initparam728549400967
7521997.rsp

./CDB20_IAD3CV:
total 4
drwxr-x--- 3 oracle oinstall 4096 Apr  8 11:49 archivelog

./CDB20_IAD3CV/archivelog:
total 4
drwxr-x--- 2 oracle oinstall 4096 Apr  8 11:50 2020_04_08

./CDB20_IAD3CV/archivelog/2020_04_08:
total 391288
-rw-r----- 1 oracle oinstall      7168 Apr  8 11:50 o1_mf_1_16_h8vgm
8xs_.arc
-rw-r----- 1 oracle oinstall      2560 Apr  8 11:50 o1_mf_1_15_h8vgm
2op_.arc
-rw-r----- 1 oracle oinstall 400666624 Apr  8 11:49 o1_mf_1_14_h8vgm
1st_.arc

./labs:
total 36
-rw-r--r-- 1 oracle oinstall 6075 Apr  8 10:11 hr_main.log
...
$ exit
SQL> EXIT
$
```

Enhanced SQL Set Operators

The SQL set operators now support all keywords as defined in ANSI SQL. The new operator `EXCEPT [ALL]` is functionally equivalent to `MINUS [ALL]`. The operators `MINUS` and `INTERSECT` now support the keyword `ALL`.

Full ANSI compliance provides greater compatibility with other database vendors and makes migration to Oracle Database easier than before.

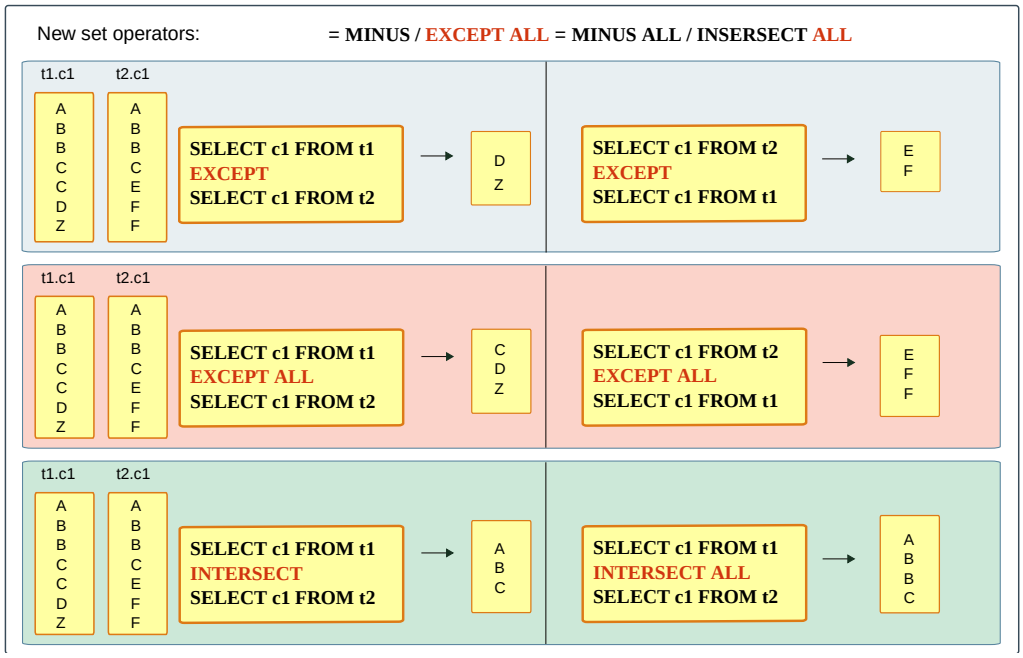
- Details: Enhanced SQL Set Operators
  This page provides more detailed information about the new SQL set operator and the enhanced existing ones.
- Practice: Using New Set Operators
  This practice shows how to use the new set operators, `EXCEPT`, `EXCEPT ALL` and `INTERSECT ALL`.

Related Topics

- *Oracle® Database Data Warehousing Guide*

Details: Enhanced SQL Set Operators

This page provides more detailed information about the new SQL set operator and the enhanced existing ones.



Until Oracle Database 20c, only the set operator `UNION` could be combined with `ALL`. Oracle Database 20c introduces two set operators, `MINUS ALL` (same as `EXCEPT ALL`) and `INTERSECT ALL`.

In the examples of the graphic, the first and second statements combining results from two queries with the `EXCEPT` operator (being equivalent to `MINUS`) returns only unique rows returned by the first query but not by the second query.

The third and forth statements combining results from two queries with the `EXCEPT ALL` operator (being equivalent to `MINUS ALL`) returns only rows returned by the first query but not by the second query, even if not unique.

The fifth and sixth statements combining results from two queries with the `INTERSECT` operator returns only unique rows returned by both queries.

Practice: Using New Set Operators

This practice shows how to use the new set operators, `EXCEPT`, `EXCEPT ALL` and `INTERSECT ALL`.

1. Before starting any new practice, refer to the practices environment recommendations.

2. Execute the `/home/oracle/labs/M104783GC10/setup_oe_tables.sh` shell script. The shell script creates and loads the `OE.INVENTORIES`, `OE.ORDERS` and `OE.ORDER_ITEMS` tables.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/setup_oe_tables.sh
...
Commit complete.

Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Connect to `PDB20` as `OE`.

```
$ sqlplus oe@PDB20
SQL*Plus: Release 20.0.0.0.0 - Production on Mon Mar 16 11:32:53 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Mon Mar 16 2020 11:32:00 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0
SQL>
```

4. Count in both tables, `INVENTORIES` and `ORDER_ITEMS`, respectively the number of products available in the inventory and the number of products that customers ordered.

```
SQL> SELECT count(distinct product_id) FROM inventories;

COUNT(PRODUCT_ID)
-----------------
              208

SQL> SELECT count(distinct product_id) FROM order_items;

COUNT(PRODUCT_ID)
-----------------
              185

SQL>
```

5. How many products are in the inventory that were never ordered? Use the `EXCEPT` operator to retrieve only unique rows returned by the first query but not by the second.

```
SQL> SELECT count(*) FROM
       (SELECT product_id FROM inventories
```

```
         EXCEPT
        SELECT product_id FROM order_items);

  COUNT(*)
----------
        84

SQL>
```

6. How many products were ordered that are now missing in the inventory? The order of the queries is relevant for the result.

```
SQL> SELECT count(*) FROM
       (SELECT product_id FROM order_items
          EXCEPT
        SELECT product_id FROM inventories);

  COUNT(*)
----------
        61

SQL>
```

7. Would the usage of `ALL` in the operator defined in the query in step 5 mean anything?

```
SQL> SELECT product_id FROM inventories
          EXCEPT ALL
        SELECT product_id FROM order_items;

PRODUCT_ID
----------
      1729
      1729
      1729
      1729
      1729
      1729
      1733
      1733
      1733
      1733
      1733
      1733
      1733
      1733
      1733
...
      3502
      3502
      3502
      3502
      3502
      3503
      3503
```

```
      3503
      3503
      3503

826 rows selected.

SQL> SELECT count(*) FROM
       (SELECT product_id FROM inventories
          EXCEPT ALL
        SELECT product_id FROM order_items);

  COUNT(*)
----------
       826

SQL>
```

The result shows all rows in the `INVENTORIES` table that contain products that were never ordered all inventories. This does not mean anything relevant. The use of `ALL` in operators must be appropriate.

8. How many products that were ordered are still orderable? The statement combining the results from two queries with the `INTERSECT` operator returns only those unique rows returned by both queries.

```
SQL> SELECT count(*) FROM
       (SELECT product_id FROM inventories
           INTERSECT
        SELECT product_id FROM order_items);

  COUNT(*)
----------
       124

SQL> SELECT count(*) FROM
       (SELECT product_id FROM order_items
           INTERSECT
        SELECT product_id FROM inventories);

  COUNT(*)
----------
       124

SQL>
```

9. Would the usage of `ALL` in the operator defined in the query in step 8 mean anything?

```
SQL> SELECT count(*) FROM
       (SELECT product_id FROM order_items
           INTERSECT ALL
        SELECT product_id FROM inventories);

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

```
        286

SQL> EXIT
$
```

The result shows all rows in the `INVENTORIES` table that contain products that were ordered. This does not mean that these products were ordered from these warehouses. The query does not mean anything relevant. The use of `ALL` in operators must be appropriate.

Upgrades, Patching and Migrations

Oracle Database Utilities

Oracle Data Pump Includes and Excludes in the Same Operation

Starting with Oracle Database 20c, Oracle Data Pump can include and exclude objects in the same export or import operation.

Oracle Data Pump provides powerful, flexible inclusion and exclusion of objects for a job. Now, Oracle Data Pump commands can include both `INCLUDE` and `EXCLUDE` parameters in the same operation. By enabling greater specificity about what is being migrated, this enhancement makes it easier to migrate to Oracle Cloud, or to another on-premises Oracle Database.

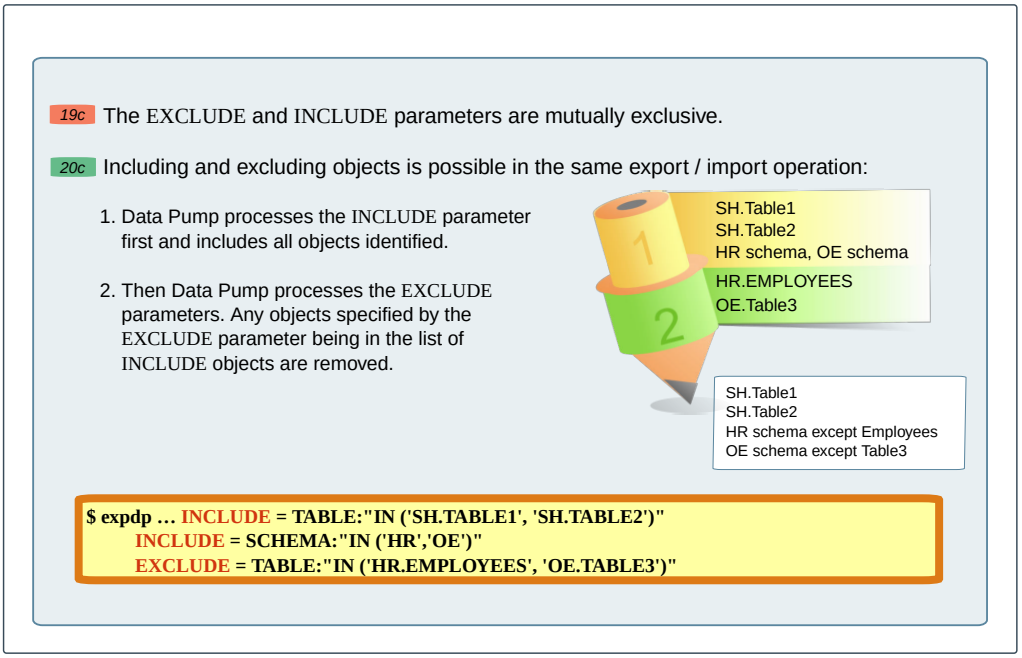
- Details: Oracle Data Pump Includes and Excludes in the Same Operation
  This page provides more detailed information about excluding and including objects with Oracle Data Pump export or import in a single command.
- Practice: Including and Excluding Objects from Export or Import
  This practice shows how to export or import objects by including and excluding objects during the same operation.

Related Topics

- *Oracle® Database Database Utilities*

Details: Oracle Data Pump Includes and Excludes in the Same Operation

This page provides more detailed information about excluding and including objects with Oracle Data Pump export or import in a single command.



Starting with Oracle Database 20c, Oracle Data Pump permits you to set both `INCLUDE` and `EXCLUDE` parameters in the same command. When you include both parameters in a command, Oracle Data Pump processes the `INCLUDE` parameter first, such that the Oracle Data Pump job includes only objects identified as included. Then it processes the `EXCLUDE` parameters, which can further restrict the objects processed by the job. As the command runs, any objects specified by the `EXCLUDE` parameter that are in the list of `INCLUDE` objects are removed.

Practice: Including and Excluding Objects from Export or Import

This practice shows how to export or import objects by including and excluding objects during the same operation.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Use the `/home/oracle/labs/M104780GC10/create_PDB20_2.sh` shell script to create the `PDB20_2` PDB and the `HR` user in `PDB20_2`.

```
$ cd /home/oracle/labs/M104780GC10
$ /home/oracle/labs/M104780GC10/create_PDB20_2.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 17 03:41:01 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER SESSION SET db_create_file_dest='/home/oracle/labs';

Session altered.
```

```
SQL> ALTER PLUGGABLE DATABASE pdb20_2 CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE pdb20_2 INCLUDING DATAFILES;

Pluggable database dropped.

SQL>
SQL> CREATE PLUGGABLE DATABASE pdb20_2
  2      ADMIN USER pdb_admin IDENTIFIED BY password ROLES=(CONNECT)
  3          DEFAULT TABLESPACE users DATAFILE SIZE 1M AUTOEXTEND ON
NEXT 1M
  4      CREATE_FILE_DEST='/home/oracle/labs';

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb20_2 OPEN;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 17 03:41:38 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP USER hr CASCADE;
DROP USER hr CASCADE
          *
ERROR at line 1:
ORA-01918: user 'HR' does not exist

SQL> CREATE USER hr IDENTIFIED BY password;

User created.

SQL> GRANT create session, create table, unlimited tablespace TO hr;

Grant succeeded.

SQL> CREATE DIRECTORY dp_dir AS '/home/oracle/labs';

Directory created.

SQL> GRANT read, write ON DIRECTORY dp_dir TO hr;
```

```
Grant succeeded.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 17 03:41:39 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:

PL/SQL procedure successfully completed.

User created.

ALTER USER hr DEFAULT TABLESPACE users
*
ERROR at line 1:
ORA-00959: tablespace 'USERS' does not exist

User altered.

Grant succeeded.

Grant succeeded.

Session altered.

Session altered.

Session altered.

******  Creating REGIONS table ....

Table created.

Index created.

Table altered.

******  Creating COUNTRIES table ....
```

```
Table created.

Table altered.

******  Creating LOCATIONS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating DEPARTMENTS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating JOBS table ....

Table created.

Index created.

Table altered.

******  Creating EMPLOYEES table ....

Table created.

Index created.

Table altered.

Table altered.

Sequence created.

******  Creating JOB_HISTORY table ....

Table created.

Index created.

Table altered.

******  Creating EMP_DETAILS_VIEW view ...

View created.
```

```
Commit complete.

Session altered.

******  Populating REGIONS table ....

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating COUNTIRES table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating LOCATIONS table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

******  Populating DEPARTMENTS table ....

Table altered.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

******  Populating JOBS table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating EMPLOYEES table ....

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating JOB_HISTORY table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

Table altered.

Commit complete.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Commit complete.

Procedure created.

Trigger created.

Trigger altered.

Procedure created.

Trigger created.

Commit complete.

Comment created.

Comment created.
```

```
Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.
```

```
Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Commit complete.

PL/SQL procedure successfully completed.

SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Before exporting the two HR tables excluding their statistics, verify that the two HR tables have statistics collected, and create a directory for the export dumpfile.
   a. Verify that the two HR tables have statistics collected.

```
$ sqlplus system@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Tue Mar 17 02:24:54
2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password: password
Last Successful login time: Tue Mar 17 2020 02:23:18 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> SELECT num_rows FROM dba_tables WHERE table_name IN
('JOBS','DEPARTMENTS');

  NUM_ROWS
----------
        27
        19
```

```
SQL>
```

b. Create a directory for the export dumpfile.

```
SQL> CREATE DIRECTORY dp_dir AS '/home/oracle/labs';

Directory created.

SQL> GRANT read, write ON DIRECTORY dp_dir TO hr;

Grant succeeded.

SQL> EXIT
$
```

4. Export from PDB20 two HR tables, excluding their statistics.

```
$ expdp hr@PDB20 DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:\"IN \
(\'JOBS\',\'DEPARTMENTS\'\)\" EXCLUDE=STATISTICS REUSE_DUMPFILES=YES
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Starting "HR"."SYS_EXPORT_SCHEMA_01":  hr/********@PDB20
DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:"IN
('JOBS','DEPARTMENTS')" EXCLUDE=STATISTICS REUSE_DUMPFILES=YES
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
. . exported "HR"."JOBS"                                  7.109 KB
19 rows
. . exported "HR"."DEPARTMENTS"                           7.125 KB
27 rows
Master table "HR"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
******
Dump file set for HR.SYS_EXPORT_SCHEMA_01 is:
  /home/oracle/labs/hr.dmp
Job "HR"."SYS_EXPORT_SCHEMA_01" successfully completed at Tue Mar 17
02:30:24 2020 elapsed 0 00:00:18
$
```

5. Import the dumpfile into another PDB, PDB20_2 in CDB20.

```
$ impdp system@PDB20_2 DUMPFILE=hr.dmp DIRECTORY=DP_DIR FULL=Y
```

```
Import: Release 20.0.0.0.0 - Production on Tue Mar 17 04:03:25 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/********@PDB20_2
DUMPFILE=hr.dmp DIRECTORY=DP_DIR FULL=Y
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "HR"."JOBS"                                 7.109 KB
19 rows
. . imported "HR"."DEPARTMENTS"                          7.125 KB
27 rows
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
ORA-39083: Object type REF_CONSTRAINT:"HR"."DEPT_LOC_FK" failed to
create with error:
ORA-00942: table or view does not exist

Failing sql is:
ALTER TABLE "HR"."DEPARTMENTS" ADD CONSTRAINT "DEPT_LOC_FK" FOREIGN KEY
("LOCATION_ID") REFERENCES "HR"."LOCATIONS" ("LOCATION_ID") ENABLE

ORA-39083: Object type REF_CONSTRAINT:"HR"."DEPT_MGR_FK" failed to
create with error:
ORA-00942: table or view does not exist

Failing sql is:
ALTER TABLE "HR"."DEPARTMENTS" ADD CONSTRAINT "DEPT_MGR_FK" FOREIGN KEY
("MANAGER_ID") REFERENCES "HR"."EMPLOYEES" ("EMPLOYEE_ID") ENABLE
...
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 19 error(s) at Tue Mar
17 04:03:37 2020 elapsed 0 00:00:05
$
```

6. The import completes with errors due to missing constraints for `HR.DEPARTMENTS` that requires constraints referring other `HR` tables. Re-execute the export operation excluding statistics and constraints.

```
$ expdp hr@PDB20 DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:\"IN \
(\'JOBS\',\'DEPARTMENTS\'\)\" EXCLUDE=STATISTICS,CONSTRAINT
REUSE_DUMPFILES=YES

Export: Release 20.0.0.0.0 - Production on Tue Mar 17 04:05:57 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
```

```
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Starting "HR"."SYS_EXPORT_SCHEMA_01":  hr/********@PDB20
DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:"IN
('JOBS','DEPARTMENTS')" EXCLUDE=STATISTICS,CONSTRAINT
REUSE_DUMPFILES=YES
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
. . exported "HR"."JOBS"                                 7.109 KB
19 rows
. . exported "HR"."DEPARTMENTS"                          7.125 KB
27 rows
Master table "HR"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
******
Dump file set for HR.SYS_EXPORT_SCHEMA_01 is:
  /home/oracle/labs/hr.dmp
Job "HR"."SYS_EXPORT_SCHEMA_01" successfully completed at Tue Mar 17
04:06:15 2020 elapsed 0 00:00:14
$
```

> **Note:**
>
> Observe that the import does not issue errors related to constraints. Constraints that should have been added to the `HR.DEPARTMENTS` table were excluded.

7. Verify that statistics for the `HR.JOBS` and `HR.DEPARTMENTS` tables were excluded too.

```
$ sqlplus system@PDB20_2

Enter password: password

SQL> SELECT num_rows FROM dba_tables WHERE table_name IN
('JOBS','DEPARTMENTS');

no  rows selected

SQL> EXIT
$
```

Oracle Data Pump Resumes Transportable Tablespace Jobs

Starting with Oracle Database 20c, Oracle Data Pump resumes transportable tablespace export and import jobs that are stopped.

Oracle Data Pump has the capacity to resume transportable tablespace export and import jobs. Due to errors, or other problems, you can find that transportable

tablespace export or import jobs are stopped. Oracle Data Pump's capacity to resume these stopped jobs helps to save you time, and makes the system more available.

Related Topics

- *Oracle® Database Database Utilities*

Oracle Data Pump Parallelizes Transportable Tablespace Metadata Operations

Starting with Oracle Database 20c, Oracle Data Pump improves Transportable Tablespace metadata operations with parallelism.

Oracle Data Pump now supports parallel export and import operations for Transportable Tablespace (TTS) metadata. This is the information that associates the tablespace data files with the target database in a TTS migration. Parallelism improves TTS export and import performance, especially when there are millions of database objects in the data files, including tables, indexes, partitions, and subpartitions.

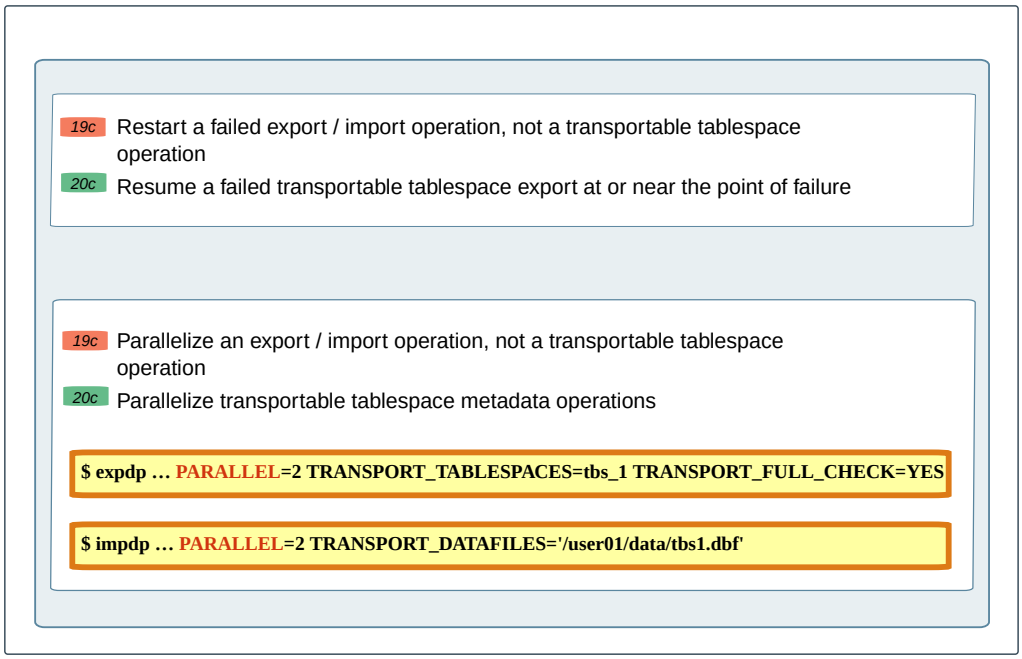
- Details: Oracle Data Pump Resumes Transportable Tablespace Jobs and Parallelizes Transportable Tablespace Metadata Operations
  This page provides more detailed information about Oracle Data Pump restartable transportable jobs and parallel export and import operations for Transportable Tablespace (TTS) metadata.
- Practice: Parallelizing TTS Metadata Operations
  The practice shows how to parallelize export and import operations for Transportable Tablespace (TTS) metadata.

Related Topics

- *Oracle® Database Database Utilities*

Details: Oracle Data Pump Resumes Transportable Tablespace Jobs and Parallelizes Transportable Tablespace Metadata Operations

This page provides more detailed information about Oracle Data Pump restartable transportable jobs and parallel export and import operations for Transportable Tablespace (TTS) metadata.



Starting with Oracle Database 20c, transportable jobs are restartable at or near the point of failure During transportable imports tablespaces are temporarily made read/ write and then set back to read-only.The temporary setting change was introduced with Oracle Database 12c Release 1 (12.1.0.2) to improve performance. However, be aware that this behavior also causes the SCNs of the import job data files to change. Changing the SCNs for data files can cause issues during future transportable imports of those files.

Practice: Parallelizing TTS Metadata Operations

The practice shows how to parallelize export and import operations for Transportable Tablespace (TTS) metadata.

1. Before starting any new practice, refer to the practices environment recommendations.
2. In the Oracle 20c PDB20, set the tablespace USERS to transport to read only.

```
$ sqlplus sys@PDB20 AS SYSDBA
SQL*Plus: Release 20.0.0.0.0 - Production on Wed Nov 20 07:29:31 2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER TABLESPACE users READ ONLY;

Tablespace altered.

SQL> EXIT
```

```
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Perform the TTS in parallel against PDB20.

```
$ expdp \"sys@PDB20 AS SYSDBA\" dumpfile=PDB20.dmp
TRANSPORT_TABLESPACES=users TRANSPORT_FULL_CHECK=YES LOGFILE=tts.log
REUSE_DUMPFILES=YES PARALLEL=2

Export: Release 20.0.0.0.0 - Production on Wed Nov 20 07:40:41 2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights
reserved.
Password: password
Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Starting "SYS"."SYS_EXPORT_TRANSPORTABLE_02":  "sys/********@PDB20 AS
SYSDBA" dumpfile=PDB20.dmp TRANSPORT_TABLESPACES=users
TRANSPORT_FULL_CHECK=YES LOGFILE=tts.log REUSE_DUMPFILES=YES PARALLEL=2
ORA-39396: Warning: exporting encrypted data using transportable option
without password

ORA-39396: Warning: exporting encrypted data using transportable option
without password

Processing object type TRANSPORTABLE_EXPORT/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TRANSPORTABLE_EXPORT/STATISTICS/TABLE_STATISTICS
Processing object type TRANSPORTABLE_EXPORT/INDEX/STATISTICS/
BITMAP_INDEX/INDEX_STATISTICS
Processing object type TRANSPORTABLE_EXPORT/PLUGTS_BLK
Processing object type TRANSPORTABLE_EXPORT/STATISTICS/MARKER
Processing object type TRANSPORTABLE_EXPORT/POST_INSTANCE/PLUGTS_BLK
Processing object type TRANSPORTABLE_EXPORT/INDEX/INDEX
Processing object type TRANSPORTABLE_EXPORT/TABLE
Processing object type TRANSPORTABLE_EXPORT/COMMENT
Processing object type TRANSPORTABLE_EXPORT/CONSTRAINT/CONSTRAINT
Processing object type TRANSPORTABLE_EXPORT/CONSTRAINT/REF_CONSTRAINT
Processing object type TRANSPORTABLE_EXPORT/TRIGGER
Processing object type TRANSPORTABLE_EXPORT/INDEX/BITMAP_INDEX/INDEX
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/
SECONDARY_TABLE/INDEX/INDEX
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/
SECONDARY_TABLE/TABLE
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/
SECONDARY_TABLE/CONSTRAINT
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/INDEX
Processing object type TRANSPORTABLE_EXPORT/MATERIALIZED_VIEW
Master table "SYS"."SYS_EXPORT_TRANSPORTABLE_02" successfully loaded/
unloaded
******************************************************************************
******
```

```
Dump file set for SYS.SYS_EXPORT_TRANSPORTABLE_02 is:
  /u01/app/oracle/admin/ORCL/dpdump/A2B63C30139C0D6BE0530600000A01C5/
PDB20.dmp
******************************************************************************
******
Datafiles required for transportable tablespace USERS:
  /u02/app/oracle/oradata/users01.dbf
Job "SYS"."SYS_EXPORT_TRANSPORTABLE_02" completed with 2 error(s) at
Wed Apr 8 13:59:55 2020 elapsed 0 00:03:36
$
```

4. Set the tablespace back to read write.

```
$ sqlplus sys@PDB20 AS SYSDBA

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Nov 20 07:29:31 2019
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> ALTER TABLESPACE users READ WRITE;

Tablespace altered.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

Oracle Data Pump Provides Optional Index Compression

In Oracle Database 20c, Oracle Data Pump supports optional index compression on imports, including for Oracle Autonomous Database.

Oracle Data Pump supports adding, changing and eliminating table compression. Oracle Database 20c supports index compression as well by introducing a new `TRANSFORM` parameter clause, `INDEX_COMPRESSION_CLAUSE`. This clause enables you to control whether index compression is performed during import. Adding this clause also enables you to specify index compression on import with the autonomous services.

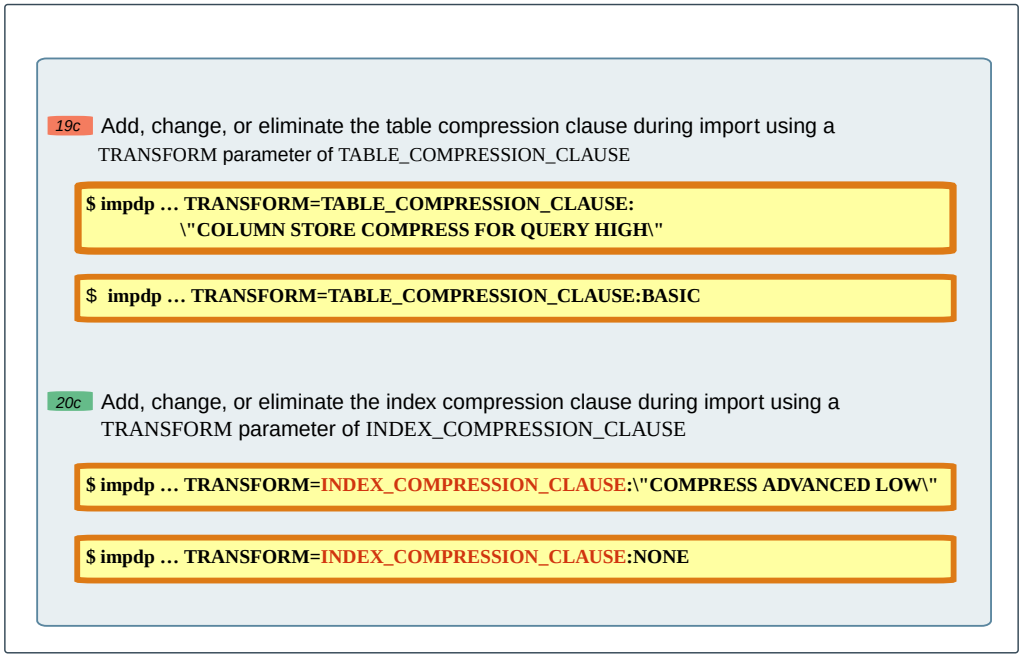
- Details: Oracle Data Pump Provides Optional Index Compression
  This page provides more detailed information about index compression during Oracle Data Pump import.
- Practice: Using Index Compression on Import
  The practice shows how to use index compression on import operations.

Related Topics

- *Oracle® Database Database Utilities*

Details: Oracle Data Pump Provides Optional Index Compression

This page provides more detailed information about index compression during Oracle Data Pump import.



If `NONE` is specified, then the index compression clause is omitted (and the index is given the default compression for the tablespace). However, if you use compression, then Oracle recommends that you use `COMPRESS ADVANCED LOW`. Indexes are created with the specified compression.

If the index compression clause is more than one word, then it must be contained in single or double quotation marks. Also, your operating system can require you to enclose the clause in escape characters, such as the backslash character. For example:

```
TRANSFORM=INDEX_COMPRESSION_CLAUSE:\"COMPRESS ADVANCED LOW\"
```

Specifying this transform changes the type of compression for all indexes in the job.

Practice: Using Index Compression on Import

The practice shows how to use index compression on import operations.

1. Before starting any new practice, refer to the practices environment recommendations.

2. Create the HR schema. Change the string *password* in the command by your password.

```
$ $ORACLE_HOME/bin/sqlplus "sys@PDB20 AS SYSDBA" @/u01/app/oracle/
product/20.0.0/dbhome_1/demo/schema/human_resources/hr_main.sql
password users temp /tmp

...
Commit complete.

PL/SQL procedure successfully completed.

SQL>
```

3. Verify that the HR.EMPLOYEES table is not using compression and does own indexes that are not using compression.

```
 SQL> CONNECT SYSTEM@PDB20
Enter password: password
Connected.
SQL> SELECT compression, compress_for FROM DBA_TABLES WHERE
table_name='EMPLOYEES';

COMPRESS COMPRESS_FOR
-------- ------------------------------
DISABLED

SQL> COL INDEX_NAME FORMAT A30
SQL> SELECT index_name, compression FROM dba_indexes WHERE
table_name='EMPLOYEES';

INDEX_NAME                     COMPRESSION
------------------------------ -------------
EMP_NAME_IX                    DISABLED
EMP_EMAIL_UK                   DISABLED
EMP_EMP_ID_PK                  DISABLED
EMP_DEPARTMENT_IX              DISABLED
EMP_JOB_IX                     DISABLED
EMP_MANAGER_IX                 DISABLED

6 rows selected.

SQL>
```

4. Create a directory for Oracle Data Pump dumpfiles.

```
SQL> CREATE DIRECTORY dp_dir AS '/u01/app/oracle/admin';

Directory created.

SQL> GRANT read, write ON DIRECTORY dp_dir TO hr;

Grant succeeded.
```

```
SQL> EXIT
$
```

5. Export the HR.EMPLOYEES table. Ignore any Database Vault warning.

```
$ expdp hr@PDB20 DUMPFILE=PDB20.dmp DIRECTORY=dp_dir TABLES=EMPLOYEES
REUSE_DUMPFILES=YES

Export: Release 20.0.0.0.0 - Production on Wed Apr 8 16:27:21 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Starting "HR"."SYS_EXPORT_TABLE_01":  hr/********@PDB20
DUMPFILE=PDB20.dmp DIRECTORY=dp_dir TABLES=EMPLOYEES REUSE_DUMPFILES=YES
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
. . exported "HR"."EMPLOYEES"                            17.08 KB
107 rows
ORA-39173: Encrypted data has been stored unencrypted in dump file set.
Master table "HR"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for HR.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/admin/PDB20.dmp
Job "HR"."SYS_EXPORT_TABLE_01" successfully completed at Wed Apr 8
16:27:55 2020 elapsed 0 00:00:29

$
```

6. Drop the table in PDB20.

```
$ sqlplus SYSTEM@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Apr 8 16:28:45 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Wed Apr 08 2020 16:24:56 +00:00

Connected to:
```

```
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
$
```

7. Import the table using the index compression and the table compression parameters.

```
$ impdp hr@PDB20 FULL=Y DUMPFILE=PDB20.dmp DIRECTORY=dp_dir
TRANSFORM=TABLE_COMPRESSION_CLAUSE:\"COMPRESS BASIC\"
TRANSFORM=INDEX_COMPRESSION_CLAUSE:\"COMPRESS ADVANCED LOW\"
EXCLUDE=CONSTRAINT
Import: Release 20.0.0.0.0 - Production on Wed Apr 8 16:39:13 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release 20.0.0.0.0
- Production
Master table "HR"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "HR"."SYS_IMPORT_FULL_01":  hr/********@PDB20 FULL=Y
DUMPFILE=PDB20.dmp DIRECTORY=dp_dir
TRANSFORM=TABLE_COMPRESSION_CLAUSE:"COMPRESS BASIC"
TRANSFORM=INDEX_COMPRESSION_CLAUSE:"COMPRESS ADVANCED LOW"
EXCLUDE=CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "HR"."EMPLOYEES"                            17.08 KB
107 rows
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
ORA-39083: Object type INDEX:"HR"."EMP_EMP_ID_PK" failed to create with
error:
ORA-25193: cannot use COMPRESS option for a single column key

Failing sql is:
CREATE UNIQUE INDEX "HR"."EMP_EMP_ID_PK" ON "HR"."EMPLOYEES"
("EMPLOYEE_ID") PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPRESS ADVANCED
LOW  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT) TABLESPACE "USERS"

Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/TRIGGER
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "HR"."SYS_IMPORT_FULL_01" completed with 1 error(s) at Wed Apr 8
```

```
16:39:55 2020 elapsed 0 00:00:36

$
```

Ignore the errors.

8. Verify that the table imported is using compression and that its indexes use compression too.

```
$ sqlplus SYSTEM@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Wed Apr 8 16:40:59 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Last Successful login time: Wed Apr 08 2020 16:38:57 +00:00

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> SELECT compression, compress_for FROM DBA_TABLES WHERE
table_name='EMPLOYEES';

COMPRESS COMPRESS_FOR
-------- ------------------------------
ENABLED  BASIC

SQL> COL INDEX_NAME FORMAT A30
SQL> SELECT index_name, compression FROM dba_indexes WHERE
table_name='EMPLOYEES';

INDEX_NAME                     COMPRESSION
------------------------------ -------------
EMP_DEPARTMENT_IX              ADVANCED LOW
EMP_JOB_IX                     ADVANCED LOW
EMP_MANAGER_IX                 ADVANCED LOW
EMP_NAME_IX                    ADVANCED LOW

SQL> EXIT
$
```

Oracle Data Pump Checksums Support Cloud Migrations

To check Oracle Data Pump dumpfiles for validity, you can now use checksums that are added to the dumpfile.

Oracle Data Pump is used for migrating application data from on-premises Oracle Database instances into the Oracle Cloud, and also for copying dumpfiles to on-premises.

Starting with Oracle Database 20c, a checksum is now added to the dumpfile. You can use the checksum to help to confirm that the file is valid after a transfer to or from the

object store and also after saving dumpfiles on on-premises and that it has no accidental or malicious changes.

- Practice: Checking Oracle Data Pump Dump Files for Validity
  This practice shows how to use the checksum to confirm that an Oracle Data Pump dump file is valid after a transfer to or from the object store and also after saving dump files on on-premises.The checksum ensures that no accidental or malicious changes occurred.

Related Topics

- *Oracle® Database Database Utilities*

Practice: Checking Oracle Data Pump Dump Files for Validity

This practice shows how to use the checksum to confirm that an Oracle Data Pump dump file is valid after a transfer to or from the object store and also after saving dump files on on-premises.The checksum ensures that no accidental or malicious changes occurred.

1. Before starting any new practice, refer to the practices environment recommendations.
2. Before starting the practice, execute the `/home/oracle/labs/M104786GC10/DP.sh` shell script. The shell script creates the table `HR.EMPLOYEES` to export in `PDB20`.

```
$ cd /home/oracle/labs/M104786GC10
$ /home/oracle/labs/M104786GC10/DP.sh
SQL*Plus: Release 20.0.0.0.0 - Production on Thu Feb 6 06:57:22 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

specify password for HR as parameter 1:

specify default tablespeace for HR as parameter 2:

specify temporary tablespace for HR as parameter 3:

specify log path as parameter 4:

PL/SQL procedure successfully completed.

User created.

User altered.

User altered.

Grant succeeded.

Grant succeeded.
```

```
Session altered.

Session altered.

Session altered.

******  Creating REGIONS table ....

Table created.

Index created.

Table altered.

******  Creating COUNTRIES table ....

Table created.

Table altered.

******  Creating LOCATIONS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating DEPARTMENTS table ....

Table created.

Index created.

Table altered.

Sequence created.

******  Creating JOBS table ....

Table created.

Index created.

Table altered.

******  Creating EMPLOYEES table ....

Table created.

Index created.

Table altered.
```

```
Table altered.

Sequence created.

******  Creating JOB_HISTORY table ....

Table created.

Index created.

Table altered.

******  Creating EMP_DETAILS_VIEW view ...

View created.

Commit complete.

Session altered.

******  Populating REGIONS table ....

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating COUNTIRES table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating LOCATIONS table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating DEPARTMENTS table ....

Table altered.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

******  Populating JOBS table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

******  Populating EMPLOYEES table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.
```

```
1 row created.

1 row created.
******  Populating JOB_HISTORY table ....

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

Table altered.

Commit complete.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Index created.

Commit complete.

Procedure created.
```

```
Trigger created.

Trigger altered.

Procedure created.

Trigger created.

Commit complete.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.
```

```
Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Comment created.

Commit complete.

SQL> Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL*Plus: Release 20.0.0.0.0 - Production on Fri Feb 7 05:24:12 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0


Directory created.

Grant succeeded.
```

```
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

3. Export the table HR.EMPLOYEES and add a checksum to the dump file to be able to confirm that the dump file is still valid after the export and that the data is intact and has not been corrupted. An Oracle Data Pump export writes control information into the header block of a dump file: Oracle Database 20c extends the data integrity checks by adding an additional checksum for all the remaining blocks beyond the header within Oracle Data Pump and external table dump files.

   a. Use the CHECKSUM parameter during the export operation.

```
$ expdp system@PDB20 TABLES=hr.employees DUMPFILE=dp_dir:emp.dmp
CHECKSUM=yes REUSE_DUMPFILES=yes

Export: Release 20.0.0.0.0 - Production on Thu Feb 6 07:14:45 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Starting "SYSTEM"."SYS_EXPORT_TABLE_01":  system/********@PDB20
TABLES=hr.employees dump file=dp_dir:emp.dmp CHECKSUM=YES
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/
TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
. . exported "HR"."EMPLOYEES"                            17.08
KB     107 rows
ORA-39173: Encrypted data has been stored unencrypted in dump file
set.
Master table "SYSTEM"."SYS_EXPORT_TABLE_01" successfully loaded/
unloaded
Generating checksums for dump file set
******************************************************************
**********
Dump file set for SYSTEM.SYS_EXPORT_TABLE_01 is:
  /home/oracle/labs/M104786GC10/emp.dmp
Job "SYSTEM"."SYS_EXPORT_TABLE_01" successfully completed at Thu
Feb 6 07:15:15 2020 elapsed 0 00:00:26
$
```

The checksum algorithm defaults to SHA256 256-bit.

b. f you want to use the SHA384 384-bit hash algorithm or SHA512 512-bit hash algorithm or the CRC32 32-bit checksum, use the CHECKSUM_ALGORITHM parameter and not the CHECKSUM parameter which uses the SHA256 256-bit hash algorithm.

```
$ expdp system@PDB20 TABLES=hr.employees DUMPFILE=dp_dir:emp384.dmp
CHECKSUM_ALGORITHM=SHA384 CHECKSUM=no REUSE_DUMPFILES=yes

Export: Release 20.0.0.0.0 - Production on Thu Feb 6 07:14:45 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
ORA-39002: invalid operation
ORA-39050: parameter CHECKSUM=NO is incompatible with parameter
CHECKSUM_ALGORITHM

$


$ expdp system@PDB20 TABLES=hr.employees DUMPFILE=dp_dir:emp512.dmp
CHECKSUM_ALGORITHM=SHA512 REUSE_DUMPFILES=yes

Export: Release 20.0.0.0.0 - Production on Thu Feb 6 07:50:05 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Starting "SYSTEM"."SYS_EXPORT_TABLE_01":  system/********@PDB20
TABLES=hr.employees dump file=dp_dir:emp512.dmp
CHECKSUM_ALGORITHM=SHA512
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/
TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
. . exported "HR"."EMPLOYEES"                            17.08
KB     107 rows
```

```
ORA-39173: Encrypted data has been stored unencrypted in dump file
set.
Master table "SYSTEM"."SYS_EXPORT_TABLE_01" successfully loaded/
unloaded
Generating checksums for dump file set
******************************************************************
**********
Dump file set for SYSTEM.SYS_EXPORT_TABLE_01 is:
  /home/oracle/labs/M104786GC10/emp512.dmp
Job "SYSTEM"."SYS_EXPORT_TABLE_01" successfully completed at Thu
Feb 6 07:46:51 2020 elapsed 0 00:00:09
$
```

4. Drop the table before importing it.

```
$ sqlplus hr@PDB20

SQL*Plus: Release 20.0.0.0.0 - Production on Thu Feb 6 08:09:49 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 - Production
Version 20.2.0.0.0

SQL> DROP TABLE employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$
```

5. Before importing the table, verify whether the dump files are corrupted or not.
   a. Corrupt one of the dump files by executing the `/home/oracle/labs/M104786GC10/corrupt.sh` shell script.

```
$ /home/oracle/labs/M104786GC10/corrupt.sh
$
```

   b. Find which of the two dump files is corrupted.

```
$ impdp system@PDB20 FULL=yes DUMPFILE=dp_dir:emp512.dmp
VERIFY_ONLY=YES

Import: Release 20.0.0.0.0 - Production on Thu Feb 6 07:21:37 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
```

```
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Verifying dump file checksums
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/
unloaded
dump file set is complete
verified checksum for dump file "/home/oracle/labs/M104786GC10/
emp512.dmp"
dump file set is consistent
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed at Fri Feb
7 05:42:40 2020 elapsed 0 00:00:01

$


$ impdp system@PDB20 FULL=yes DUMPFILE=dp_dir:emp.dmp
VERIFY_ONLY=YES

Import: Release 20.0.0.0.0 - Production on Thu Feb 6 07:21:37 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password
Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39411: header checksum error in dump file "/home/oracle/labs/
M104786GC10/emp.dmp"

$ oerr ora 39411
39411, 00000, "header checksum error in dump file \"%s\""
// *Cause:  The header block for the Data Pump dump file contained a
//          header checksum that did not match the value calculated
from the
//          header block as read from disk. This indicates that the
header
//          was tampered with or otherwise corrupted due to
transmission or
//          media failure.
// *Action: Contact Oracle Support Services.
$
```

6. Import the table.
   a. Import the table using the corrupted dump file. If checksums were generated when the export dump files were completed, the checksum is verified during the import.

```
$ impdp system@PDB20 FULL=yes DUMPFILE=dp_dir:emp.dmp

Import: Release 20.0.0.0.0 - Production on Tue Mar 17 07:19:24 2020
```

```
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39411: header checksum error in dump file "/home/oracle/labs/
M104786GC10/emp.dmp"

$
```

b. Import the table using the non-corrupted dump file. If checksums were generated when the export dump files were completed, the checksum is verified during the import if you mention the parameter `VERIFY_CHECKSUM`. Ignore the error messages related to indexes creation. The important in this practice is that the table can be reimported.

```
$ impdp system@PDB20 FULL=yes DUMPFILE=dp_dir:emp512.dmp
VERIFY_CHECKSUM=YES

Import: Release 20.0.0.0.0 - Production on Thu Feb 6 09:48:44 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password

Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Verifying dump file checksums
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/
unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/********@PDB20
FULL=yes DUMPFILE=dp_dir:emp512.dmp VERIFY_CHECKSUM=YES
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "HR"."EMPLOYEES"                            17.08
KB     107 rows
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
Processing object type TABLE_EXPORT/TABLE/STATISTICS/
TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed at Tue Mar
17 07:20:29 2020 elapsed 0 00:00:20
```

```
$
```

c. Import using the non-corrupted dumpfile avoiding the verification. Drop the table first.

```
$ sqlplus hr@pdb20

SQL*Plus: Release 20.0.0.0.0 - Production on Thu Feb 6 08:09:49 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter password: password

Connected to:
Oracle Database 20c Enterprise Edition Release 20.0.0.0.0 -
Production
Version 20.2.0.0.0

SQL> DROP TABLE employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
Disconnected from Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Version 20.2.0.0.0
$ impdp hr@PDB20 FULL=yes DUMPFILE=dp_dir:emp512.dmp
VERIFY_CHECKSUM=NO

Import: Release 20.0.0.0.0 - Production on Thu Feb 6 07:21:37 2020
Version 20.2.0.0.0

Copyright (c) 1982, 2020, Oracle and/or its affiliates.  All rights
reserved.
Password: password
Master table "HR"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Connected to: Oracle Database 20c Enterprise Edition Release
20.0.0.0.0 - Production
Warning: dump file checksum verification is disabled
Master table "HR"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "HR"."SYS_IMPORT_FULL_01":  system/********@PDB20 FULL=yes
DUMPFILE=dp_dir:emp512.dmp VERIFY_CHECKSUM=NO
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "HR"."EMPLOYEES"                            17.08 KB
107 rows
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/
INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
```

```
Processing object type TABLE_EXPORT/TABLE/STATISTICS/
TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "HR"."SYS_IMPORT_FULL_01" successfully completed at Tue Mar 17
07:22:04 2020 elapsed 0 00:00:20
$
```