

Oracle® Database

Learning Database New Features



F36089-15
August 2021

ORACLE®

Copyright © 2021, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Table of Contents

Table of Contents	2
Preface	18
Audience	18
Documentation Accessibility	18
Access to Oracle Support	18
Related Documents	18
Conventions	18
Learning Database New Features	20
Practices	21
Database Features and Licensing App	21
Database 21c Interactive Architecture Diagram	21
Practices Environment	23
Overview	23
Step 1 : Create an Oracle Database 21c instance	23
Step 2 : Define and test the connections	23
Step 3 : Download the practices scripts	27
Step 4 : Update the practices scripts to the current environment	28
Application Development	30
Application Express	31
Application Backups	32
Related Topics	32
Dark Mode	33
Related Topics	33
Export App as Zip	34
Related Topics	34
Faceted Search	35
Related Topics	35
Friendly URL Syntax	36
Related Topics	36
Mega Menus	37
Related Topics	37
Native PDF Printing	38
Related Topics	38
New Data Upload	39
Related Topics	39
New Form Region	40
Related Topics	40
New Team Development	41
Related Topics	41
Oracle JET Charts	42
Related Topics	42
Popup List of Values (LOV)	43
Redwood UI	44

Remote Application Deployment	45
Related Topics	45
REST Read / Write Enhancements	46
Related Topics	46
Shared List of Values (LOV)	47
Related Topics	47
SQL Workshop Now Supports Simple Oracle Document Access (SODA)	48
Related Topics	48
Universal Theme Enhancements	49
Related Topics	49
Globalization	50
Enhanced Parallel Execution Applicability	51
New German Linguistic Sorts for Capital Sharp S Support	52
Related Topics	52
New Era Support for Japanese Imperial Calendar	53
Related Topics	53
Unicode 12.1 Support	54
Related Topics	54
Zero Downtime Upgrade of Timezone Data	55
Related Topics	55
Java in Oracle Database	56
Oracle JVM Security Enhancements	57
Related Topics	57
JDBC	58
Java Library for Reactive Streams Ingestion	59
Related Topics	59
JDBC DataSource for Sharded Databases Access	60
Related Topics	60
JDBC Reactive Extensions	61
Related Topics	61
JDBC Support for Native JSON Data Type	62
Related Topics	62
JSON Document Store	63
New JSON Data Type	64
Related Topics	64
New Oracle SQL Function JSON_TRANSFORM	65
Related Topics	65
SQL/JSON Syntax Improvements	66
Related Topics	66
Multivalue Index for JSON	67
Related Topics	67
Enhancements to View Creation and of Virtual Columns Addition	68
Related Topics	68
JSON Scalar Allowed at Top Level of JSON Document (RFC 8259 Support)	69
Related Topics	69
Compatibility for Old Clients With Binary JSON	70
Related Topics	70
Net Services	71

Migration of Oracle CMAN Sessions with High Availability	72
Related Topics	72
Oracle CMAN Traffic Director Mode Support for All Types of Database Links	73
Related Topics	73
REST APIs for Oracle CMAN Administration, Proxy Protocol, Enhanced Rule List, and Bandwidth Management	7474
Related Topics	7474
Reverse Connection Support Using CMAN Tunnels	75
Related Topics	75
Details: Reverse Connection Support Using CMAN Tunnels	75
Oracle Call Interface	77
Improvements to OCI Data Interface for LOBs	78
Related Topics	78
New C Client Interface APIs for JSON Data Type	79
Related Topics	79
Oracle Call Interface API to build and quote TNS Connection Strings	80
Related Topics	80
Oracle Call Interface Session Pool Improvements	81
Related Topics	81
Oracle SODA for C and PL/SQL APIs Enhancements	82
Related Topics	82
Support for C99 Syntax in Pro*C/C++ Precompiler	83
Related Topics	83
Oracle Data Provider for .NET (ODP.NET)	84
Managed ODP.NET and ODP.NET Core: Bulk Copy	85
Managed ODP.NET and ODP.NET Core: Debug Tracing Redaction	86
Managed ODP.NET and ODP.NET Core: WebSocket and WebSocket Secure	87
PL/SQL	88
DBMS_CLOUD Package	89
Related Topics	89
JavaScript Execution using DBMS_MLE	90
Related Topics	90
New PL/SQL Iterator Constructs	91
Related Topics	91
New Pragma SUPPRESSES_WARNING_6009	92
Related Topics	92
PL/SQL Qualified Expressions Enhancements	93
Related Topics	93
PL/SQL Support For New JSON SQL Data Type	94
Related Topics	94
PL/SQL Type Attributes in User-Defined Types	95
Related Topics	95
SQL	96
Enhanced SQL Set Operators	97
Related Topics	97
Details: Enhanced SQL Set Operators	97
Practice: Using New Set Operators	98
Overview	98
Step 1: Set up the environment	99

Step 2 : Test the set operator with the EXCEPT clause	99
Step 3: Test the set operator with the EXCEPT ALL clause	100
Step 4 : Test the set operator with the INTERSECT clause	102
Step 5 : Test the set operator with the INTERSECT ALL clause	102
Expression Support for Initialization Parameters	104
Related Topics	104
Practice: Using Expressions in Initialization Parameters	104
Overview	104
Step 1: Test	104
Placeholders in SQL DDL Statements	109
Related Topics	109
Details: Placeholders in SQL Statements	109
SQL Macros	111
Related Topics	111
Details: SQL Macros	111
Practice: Using SQM Scalar and Table Expressions	114
Overview	114
Step 1 : Use SQL Macros as a scalar expression	114
Step 2 : Use SQL Macros as a table expression	117
Text	122
Custom Range Bucketing in Result Set Interface	123
Related Topics	123
Facet Navigation Support for JSON Search Indexes	124
Related Topics	124
Improved Index Synchronization and Automatic Index Optimization	125
Related Topics	125
In-Memory Full Text Columns	126
Related Topics	126
JSON Support in Result Set Interface	127
Related Topics	127
Named Entity Recognition Improvements	128
Related Topics	128
New DIRECTORY_DATASTORE Data Store Type for Oracle Text	129
Related Topics	129
New NETWORK_DATASTORE Data Store Type for Oracle Text	130
Related Topics	130
New Oracle Text Index Type: Search Index	131
Related Topics	131
Big Data and Data Warehousing Solutions	132
Analytic Views	133
Analytic View Enhancements to SQL and PL/SQL	134
Related Topics	134
Analytical SQL and Statistical Functions	135
Bitwise Aggregate Functions	136
Related Topics	136
Practice: Using Bitwise Aggregate Functions	136
Overview	136
Step 1: Test the bitwise AND function	136

Step 2 : Test the bitwise OR function	137
Step 3 : Test the bitwise XOR function	137
Enhanced Analytic Functions	139
Related Topics	139
Practice: Using Enhanced Analytic Functions	139
Overview	139
Step 1 : Set up the environment	139
Step 2 : Experiment with the GROUPS clause of the window frame	141
Step 3 : Experiment the usage of the EXCLUDE clause of the window frame	145
Step 4 : Experiment with the GROUPS and EXCLUDE clauses of the window frame	148
New Analytical and Statistical Aggregate Functions	150
Related Topics	150
Practice: Detecting Data Tampering with the CHECKSUM Function	151
Overview	151
Step 1 : Set up the environment	151
Step 2 : Examine data before tampering	152
Step 3 : Examine data after tampering	154
Practice: Measuring Asymmetry in Data with the SKEWNESS Functions	159
Overview	159
Step 1 : Set up the environment	159
Step 2 : Examine skewed data	160
Step 3 : Examine skewed data after data evolution	162
Practice: Measuring Tailedness of Data with the KURTOSIS Functions	165
Overview	166
Step 1 : Set up the environment	166
Step 2 : Examine the kurtosis of the distribution	167
Step 3 : Examine the kurtosis of the distribution after data evolution	169
Machine Learning for SQL	173
Adam Optimization Solver for the Neural Network Algorithm	174
Related Topics	174
Oracle Machine Learning MSET-SPRT Algorithm	175
Related Topics	175
Oracle Machine Learning XGBoost Algorithm	176
Related Topics	176
Machine Learning for Python	177
Oracle Machine Learning for Python (OML4Py)	178
Related Topics	178
Details: AutoML in OML4Py	178
Oracle Machine Learning for Python Configuration in DBCA	180
Related Topics	180
Query Optimization	181
In-Memory Vectorized Joins	182
Related Topics	182
Details: In-Memory Deep Vectorization	182
Spatial and Graph	184
Property Graph: Graph Server and Client Kit	185
Related Topics	185
Property Graph: Native Python Client	186

Related Topics	186
Property Graph: New Features in PGQL	187
Related Topics	187
Property Graph: Optimized Graph Representation for Faster Performance	188
Related Topics	188
Property Graph: User-defined Graph Algorithms	189
Related Topics	189
Property Graph Visualization	190
Related Topics	190
RDF Graph: Native Unicode Storage and Processing	191
Related Topics	191
Self-service Low-code Spatial Studio	192
Related Topics	192
Spatial Network Data Model Contraction Hierarchy	193
Related Topics	193
Spatial Support for Database In-Memory	194
Related Topics	194
Database Upgrade and Utilities	195
Database Utilities	196
Oracle Data Pump Checksums Support Cloud Migrations	197
Related Topics	197
Practice: Checking Oracle Data Pump Dump Files for Validity	197
Overview	197
Step 1 : Set up the environment	197
Step 2 : Export the table using the checksum	199
Step 3 : Import the table	201
Oracle Data Pump Exports from Oracle Autonomous Database	205
Related Topics	205
Oracle Data Pump Includes and Excludes in the Same Operation	206
Related Topics	206
Details: Oracle Data Pump Includes and Excludes in the Same Operation	206
Practice: Including and Excluding Objects from Export or Import	207
Overview	207
Step 1 : Set up the environment	207
Step 2 : Export tables excluding their statistics	209
Step 3 : Import tables	210
Oracle Data Pump Parallelizes Transportable Tablespace Metadata Operations	215
Related Topics	215
Details: Oracle Data Pump Resumes Transportable Tablespace Jobs and Parallelizes Transportable Tablespace Metadata Operations	215
Practice: Parallelizing TTS Metadata Operations	216
Overview	216
Step 1 : Prepare the tablespace to be exported	216
Step 2 : Perform the TTS in parallel	217
Step 3 : Set the tablespace back to read write	218
Oracle Data Pump Provides Optional Index Compression	220
Related Topics	220
Details: Oracle Data Pump Provides Optional Index Compression	220

Practice: Using Index Compression on Import	221
Overview	221
Step 1 : Set up the environment	221
Step 2 : Export the table	223
Step 3 : Import the table using the compression parameters	224
Oracle Data Pump Resumes Transportable Tablespace Jobs	227
Related Topics	227
Oracle Data Pump Supports Export to and Import From Cloud Object Stores	228
Related Topics	228
Oracle Data Pump Supports Native JSON Datatypes	229
Oracle SQL*Loader Support for Object Store Credentials	230
Related Topics	230
Oracle SQL*Loader Supports Native JSON Data Type	231
Related Topics	231
Upgrades and Migration	232
AutoUpgrade Automates Data Guard Operations During Database Upgrade	233
Related Topics	233
AutoUpgrade Automates Steps Required for Oracle RAC Database Upgrade	234
Related topics	234
AutoUpgrade Automates Upgrade and Conversion of Non-CDB to PDB	235
Related Topics	235
AutoUpgrade Automates Upgrade of a PDB via Unplug-Plug-Upgrade	236
Related Topics	236
Oracle Database Automates Database Upgrades with AutoUpgrade	237
Related Topics	237
Management Solutions	238
Diagnosability	239
Enhanced Diagnosability of Oracle Database	240
Related Topics	240
SQL*Net Improved Diagnosability	241
Related Topics	241
Manageability	242
Persistent Memory Database	243
Related Topics	243
DAX-Enabled File Systems	244
Related Topics	244
New Database Initialization Parameters for Database Resident Connection Pooling (DRCP)	245
Related Topics	245
Multi-Mount DBFS Client	246
Related Topics	246
Near Zero Brownout for Planned Maintenance	247
Related Topics	247
Details: Near Zero Brownout for Planned Maintenance	247
Oracle Grid Infrastructure SwitchHome	249
Related Topics	249
Read-Only Oracle Home Default	250
Related Topics	250
Performance and High Availability	251

ACFS Cluster File System	252
AutoShrink for ACFS	253
Related Topics	253
Details: Oracle ACFS Automatic Shrinking	253
Mixed Sector Support	255
Related Topics	255
Details: Mixed Sector Support	255
Oracle ACFS File Based Snapshots	256
Related Topics	256
Details: Oracle ACFS File Based Snapshots	256
Replication Unplanned Failover	258
Related Topics	258
Details: Oracle ACFS Replication Unplanned Failover	258
Application Continuity	260
Application Continuity Protection Check	261
Related Topics	261
Planned Failover	262
Related Topics	262
Reset Session State	263
Related Topics	263
Transparent Application Continuity	264
Related Topics	264
Transparent Application Continuity in the Oracle Cloud	265
Related Topics	265
Automatic Operations	266
Automatic Indexing Enhancements	267
Related Topics	267
Automatic Index Optimization	268
Related Topics	268
Details: Automatic Index Optimization	268
Practice: Implementing Storage Tiering ADO Policy for Indexes	270
Overview	270
Step 1 : Set up the environment for testing	270
Step 2 : Display the space used and freed by the table index in the tablespace	271
Step 3 : Create a storage tiering ADO policy on the index	273
Step 4 : Test the storage tiering ADO policy	274
Step 5 : Drop the ADO policy	279
Automatic Materialized Views	281
Related Topics	281
Automatic SQL Tuning Set	282
Related Topics	282
Automatic Temporary Tablespace Shrink	283
Automatic Undo Tablespace Shrink	284
Automatic Zone Maps	285
Related Topics	285
Details: Automatic Zone Maps	285
Details: Automatic Zone Maps - Package	286
Details: Automatic Zone Maps - Views	288

Object Activity Tracking System	290
Related Topics	290
Sequence Dynamic Cache Resizing	291
Related Topics	291
Automatic Storage Manager (ASM)	292
Enable ASMCA to Configure Flex ASM on an Existing NAS Configuration	293
Related Topics	293
Enhanced Double Parity Protection for Flex and Extended Disk Groups	294
Related Topics	294
File Group Templates	295
Related Topics	295
Oracle ASM Flex Disk Group Support for Cloning a PDB in One CDB to a New PDB in a Different CDB	296
Related Topics	296
Autonomous Health Framework	297
Enhanced Support for Oracle Exadata	298
Related Topics	298
Oracle Cluster Health Advisor Support for Solaris	299
Related Topics	299
Oracle Cluster Health Monitor Local Mode Support	300
Related Topics	300
Oracle ORAchk and EXAchk Support for REST API	301
Related Topics	301
Oracle Trace File Analyzer Real-Time Health Summary	302
Related Topics	302
Oracle Trace File Analyzer Support for Efficient Multiple Service Request Data Collections	303
Related Topics	303
Remote GIMR Support for Oracle Standalone Clusters	304
Related Topics	304
Support for Automatically Enabling Oracle Database Quality of Service (QoS) Management	305
Related Topics	305
Support for Deploying Grid Infrastructure Management Repository (GIMR) into a Separate Oracle Home	306
Related Topics	306
Clusterware	307
Clusterware REST API	308
Related Topics	308
Common Data Model Across Fleet Patching and Provisioning Servers	309
Related Topics	309
Details: Fleet Patching and Provisioning Common Data Model	309
FPP Integration with AutoUpgrade	311
Related Topics	311
Details: Oracle Clusterware 21c Deprecated and Desupported Features	312
Database In-Memory	313
Database In-Memory Base Level	314
Related Topics	314
Automatic In-Memory	315
Related Topics	315
Details: Automatic In-Memory	315
Practice: Configuring and Observing Automatic In-Memory	316

Overview	316
Step 1 : Set up the environment with In-Memory Column Store	317
Step 2: Configure in-memory tables	319
Step 3 : Configure Automatic In-Memory	320
Step 4 : Test	323
Database In-Memory External Table Enhancements	329
Related Topics	329
In-Memory Hybrid Scans	330
Related Topics	330
Details: In-Memory Hybrid Scans	330
Practice: Using In-Memory Hybrid Scans in Queries	331
Overview	332
Step 1 : Set up the environment with In-Memory Column Store	332
Step 2 : Populate the in-memory table	333
Step 3 : Complete In-Memory Scans	335
Step 4 : Drop the user	339
CellMemory Level	340
Related Topics	340
Database In-Memory: Additional Features	341
Data Guard	342
Active Data Guard - Standby Result Cache	343
Related Topics	343
Data Guard Broker Far Sync Instance Creation	344
Related Topics	344
Details: Data Guard Broker Far Sync Instance Creation	344
Command Parameters	345
Data Guard Far Sync instance in Maximum Performance Mode	347
Related Topics	347
Fast-Start Failover Callouts	348
Related Topics	348
Fast-Start Failover Configuration Validation	349
Related Topics	349
PDB Recovery Isolation	350
Related Topics	350
Details: PDB Recovery Isolation	350
Standardized Data Guard Broker Directory Structure	352
Related Topics	352
Other features	353
Callout Configuration Scripts	353
Related Topics	354
Data Guard Broker Managed Default Directory	354
Permissions Required by the DG_ADMIN Directory	357
Related Topics	358
Oracle Database 21c Data Guard Desupported Features	358
The FastStartFailoverLagLimit Configuration Property	358
Related Topics	359
The PREPARE DATABASE FOR DATA GUARD command	359
Command Parameters	360

Command Example	362
Related Topics	364
The VALIDATE FAST_START FAILOVER Command	364
Related Topics	365
Flashback	366
Migrate Flashback Data Archive-Enabled Tables Between Different Database Releases	367
Related Topics	367
Flashback Database Support for Datafile Shrink	368
Related Topics	368
PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past	369
Related Topics	369
Details: PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past	369
Practice: Flashing Back PDBs to Any Time in the Recent Past	370
Overview	371
Step 1 : Set up the environment	371
Step 2 : Generate an error on a table	372
Step 3 : Restore the table	373
Step 4 : Generate a second error on a table	374
Step 5 : Restore the table back to the point before the table was dropped	376
GoldenGate	382
Automatic CDR Enhancements	383
Related Topics	383
Improved Support for Table Replication for Oracle GoldenGate	384
Related Topics	384
LogMiner Views Added to Assist Replication	385
Related Topics	385
Oracle GoldenGate for Oracle and XStream Support for JSON Data Type	386
Related Topics	386
Multitenant	387
DRCP Enhancements for Oracle Multitenant	388
Related Topics	388
Expanded Syntax for PDB Application Synchronization	389
Related Topics	389
Details: Expanded Syntax for PDB Application Synchronization	389
Practice: Synchronizing Multiple Applications In Application PDBs	390
Overview	390
Step 1 : Set up the environment	391
Step 2 : Display the installed applications	396
Step 3 : Synchronize the application PDBs	397
MAX_IDLE_BLOCKER_TIME Parameter	399
Related Topics	399
Details: MAX_IDLE_BLOCKER_TIME Parameter	399
Practice: Using the MAX_IDLE_BLOCKER_TIME Parameter	401
Overview	401
Step 1 : Set up the environment with two sessions	401
Step 2 : Set MAX_IDLE_BLOCKER_TIME to two minutes	401
Step 3 : Test	402
Step 4 : Examine the DIAG trace files	403

Namespace Integration with Database	406
Related Topics	406
Details: Database Nest	406
Support Per-PDB Capture for Oracle Autonomous Database	408
Related Topics	408
Time Zone support for PDBs in DBCA	409
Related Topics	409
Details: Using non-CDBs and CDBs	410
Oracle Real Application Clusters (RAC)	411
Cache Fusion Hardening	412
Related Topics	412
Details: Cache Fusion Hardening	412
Database Management Policy change for Oracle RAC in DBCA	414
Related Topics	414
Integration of PDB as a Resource in Clusterware	415
Related Topics	415
Pluggable Database Cluster Resources	416
Related Topics	416
SecureFiles	417
SecureFiles Shrink	418
Related Topics	418
Details: SecureFiles Defragmentation	418
Practice: Shrinking SecureFile LOBs	419
Overview	419
Step 1 : Create a table with a SecureFile LOB	419
Step 2 : Shrink the SecureFile LOB after rows are inserted and updated	420
Step 3 : Shrink the SecureFile LOB after rows are updated	422
Sharding	426
Centralized Backup and Restore of a Sharded Database	427
Related Topics	427
Create a Sharded Database from Multiple Existing Databases (Federated Sharding)	428
Related Topics	428
Multi-Shard Query, Data Loading, and DML Enhancements	429
Related Topics	429
Sharding Advisor Schema Analysis Tool	430
Related Topics	430
Transactional Event Queues (TEQs)	431
Advanced Queuing Support for JSON Data Type	432
Related Topics	432
Advanced Queuing: Kafka Java Client for Transactional Event Queues	433
Related Topics	433
Advanced Queuing: PL/SQL Enqueue and Dequeue Support for JMS Payload in Transactional Event Queues	434
Related Topics	434
Advanced Queuing: PL/SQL Enqueue and Dequeue Support for non-JMS Payload in Transactional Event Queues	435
Related Topics	435
Advanced Queuing: Simplified Metadata and Schema in Transactional Event Queues	436
Related Topics	436

Advanced Queuing: Transactional Event Queues for Performance and Scalability	437
Related Topics	437
Security Solutions	438
Advanced Security	439
Ability to Control Heartbeats in United Mode and Isolated Mode PDBs	440
Related Topics	440
Ability to Set the Default Tablespace Encryption Algorithm	441
Related Topics	441
Practice: Setting the Default Tablespace Encryption Algorithm	441
Overview	441
Step 1 : Set the default tablespace encryption algorithm	441
Step 2 : Verify the tablespace encryption algorithm used	442
Enhanced Database Availability with Zero Downtime to Switch Over to an Updated PKCS#11 Library	444
Related Topics	444
Improved Performance with Large Numbers of TDE Keys in Wallets or Oracle Key Vault	445
Sharing of TDE Master Encryption Key Across Oracle Processes	446
Related Topics	446
Database Vault	447
Ability to Prevent Local Oracle Database Vault Policies from Blocking Common Operations	448
Related Topics	448
Practice: Preventing Local Users from Blocking Common Operations - Realms	448
Overview	448
Step 1 : Configure and enable Database Vault at the CDB and PDB levels	449
Step 2 : Test table data accessibility with no realm on common objects	451
Step 3 : Test table data accessibility with a common regular or mandatory realm on common objects	453
Step 4 : Test table data accessibility on common objects with a PDB regular or mandatory realm	458
Step 5 : Restrict local users from creating Oracle Database Vault controls on common objects	463
Step 6 : Test table data accessibility with a common regular or mandatory realm on common objects	463
Step 7 : Test table data accessibility on common objects with a PDB regular or mandatory realm	468
Step 8 : Summary	474
Step 9 : Disable Database Vault in both the PDB and the CDB root	474
Practice: Preventing Local Users from Blocking Common Operations - Command Rules	475
Overview	475
Step 1 : Configure and enable Database Vault at the CDB and PDB levels	476
Step 2 : Test CDB and PDB connections with no command rule on common users	477
Step 3 : Test CDB and PDB connections with a command rule in CDB root on common users	478
Step 4 : Test CDB and PDB connections with a command rule in the PDB on common users	480
Step 5 : Prevent local users from creating Oracle Database Vault controls on common users that prevent them from logging in to the PDB	481
Step 6 : Test CDB and PDB connections with a command rule in the CDB root on common users	482
Step 7 : Test CDB and PDB connections with a command rule in a PDB on common users	484
Step 8 : Summary	485
Step 9 : Disable Database Vault in both the PDB and the CDB root	485
ADMINISTER KEY MANAGEMENT Statement Now Protected by Oracle Database Vault Command Rules	487
Related Topics	487
DBA_DV_SIMULATION_LOG View Columns REALM_NAME and RULE_SET_NAME Now VARCHAR2 Data Type	488
Related Topics	488
No Need to Disable Oracle Database Vault Before Upgrades	489

Related Topics	489
Uninstalling and Installing Oracle Label Security and Oracle Database Vault Now Supported	490
Related Topics	490
Practice: Uninstalling Oracle Database Vault	490
Overview	490
Step 1 : Ensure Database Vault is enabled before uninstalling	490
Step 2 : Disable Database Vault at the PDB and CDB levels	492
Step 3 : Remove Database Vault metadata at the PDB and CDB levels	495
Practice: Installing Oracle Database Vault	498
Overview	498
Step 1 : Ensure Database Vault is not installed	498
Step 2 : Use DBCA to reinstall or install Database Vault in the CDB	498
Security	502
Oracle Blockchain Table	503
Related Topics	503
Details: Oracle Blockchain Table	503
Practice: Managing Blockchain Tables and Rows	506
Overview	507
Step 1 : Create the blockchain table	507
Step 2 : Insert rows into the blockchain table	510
Step 3 : Delete rows from the blockchain table	513
Step 4 : Drop the blockchain table	514
Step 5 : Check the validity of rows in the blockchain table	515
Immutable Tables	518
Related Topics	518
Details: Immutable Tables	518
Why Immutable Tables?	518
Actions with Immutable Tables	518
Creating an Immutable Table	519
SQL Commands and PL/SQL Procedures for Immutable Tables	519
Practice: Immutable Tables	520
Overview	520
Step 1 : Create an Immutable Table	520
Step 2 : Insert Rows into the Immutable Table	522
Step 3 : Modify the Definition of the Immutable Table	523
Step 4 : Delete Obsolete Rows from an Immutable Table	524
Step 5: Drop the Immutable Table	525
Authentication and Authorization	526
Ability to Specify the Location of the CMU Wallet and dsi.ora File with a Database Property	526
Related Topics	526
Ability to Use Multiple Kerberos Principals with a Single Database Client	526
Addition of USER_APPLICATION_ROLES Data Dictionary View	527
Related Topics	527
Connect to Multiple Databases with Different Certificates from a Single Client	527
Related Topics	527
Enterprise User Manager Support for Per-PDB Directory Service Connections	527
Related Topics	528
Force Upgraded Password File to be Case Sensitive	528

Related Topics	528
Practice: Forcing an Upgraded Password File to be Case Sensitive	528
Overview	529
Step 1 : Display the password file format of CDB21	529
Step 2 : Change the SYS password and verify that the password is now case-sensitive	529
Gradual Database Password Rollover for Applications	532
Related Topics	533
Minimum Password Length Enforcement for All PDBs	533
Related Topics	533
Practice: Enforcing a Minimum Password Length on All PDBs	533
Overview	533
Step 1 : Create a mandatory profile in the CDB root	534
Step 2 : Set the MANDATORY_USER_PROFILE initialization parameter	535
Step 3 : Replace the password verification function to enforce the minimum password length.	536
Step 4 : Test	537
Step 5 : Reset the configuration	538
New and Updated Password User Profiles for STIG and CIS	541
Related Topics	541
Oracle Database Connections to Kerberos Servers Now Default to TCP	542
Related Topics	542
Windows Authentication No Longer Uses NTLM by Default	542
Related Topics	542
Encryption	543
Updated Support for Micro Edition Suite (MES) for FIPS 140.2	543
Related Topics	543
Support for DBMS_CRYPTO Asymmetric Key Operations	543
Related Topics	543
Audit	544
Auditing for Oracle XML DB HTTP and FTP Services	544
Related Topics	544
Predefined Unified Audit Policies for Security Technical Implementation Guide (STIG) Compliance	544
Related Topics	545
Practice: Using Predefined Unified Audit Policies for STIG Compliance	545
Overview	545
Step 1 : View the predefined unified audit policies that are implemented	545
Step 2 : Enable all three audit policies for all users	549
SYSLOG Destination for Common Unified Audit Policies	550
Related Topics	550
Practice: SYSLOG Destination for Common Unified Audit Policies	551
Overview	551
Step 1 : Create a common user	551
Step 2 : Create a common and local audit policy	552
Step 3 : Configure the SYSLOG destination for common and local audit policies	554
Step 4 : Define the OS directories for the SYSLOG files	556
Step 5 : Test	557
Step 6: Cleanup	561
Unified Audit Policies Enforced on the Current User	562
Related Topics	563

Details: Unified Audit Policies Enforced on the Current User	563
Practice: Enforcing Unified Audit Policies on the Current User	564
Overview	564
Step 1 : Create the users and a procedure	564
Step 2 : Create and enable an audit policy	566
Step 3 : Test	566
Unified Audit Policy Configuration Changes Effective Immediately	567
Related Topics	567
Details: Unified Audit Policy Configuration Changes Effective Immediately	568
Practice: Auditing Actions on Connected Sessions	568
Overview	568
Step 1 : Create the local user in PDB21	569
Step 2 : Create and enable an audit policy on the HR.LOCATIONS table	571
Step 3 : Test	572
Unified Auditing on an Editioned Object Now Applies to All Its Editions	573
Related Topics	573

Preface

This document describes new features implemented in Oracle Database 21c. Additionally, this document provides you with practices for key new features.

Audience

Read Learning Database New Features if you want to learn about features, options, and enhancements that are new in Oracle Database 21c.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <https://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 <https://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <https://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 21c documentation set:

- 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.
<i>italic</i>	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.

Learning Database New Features

This asset contains descriptions of all the Oracle Database Release 21c new features, as well as complementary details and practices for key new features.

You can listen to the presentation from Andy Mendelsohn, Executive VP of Oracle Database Server Technologies, about the core Oracle Database Breakthrough Innovations.




You can read VP of Database Product Management, [William Hardie's](#) blog about the core Oracle Database 21c features by visiting <https://blogs.oracle.com/database/introducing-oracle-database-21c>.

December 8, 2020



Introducing Oracle Database 21c



William Hardie
VICE PRESIDENT

The latest Innovation release of the world's most popular database, Oracle Database 21c, is now generally available "cloud first" in the Oracle Cloud **Database Service Virtual Machine** (for RAC and single instance) and **Bare Metal Service** (single instance). It's also available in the **Autonomous Database Free Tier Service** in Ashburn (IAD), Phoenix (PHX), Frankfurt (FRA) and London (LHR) regions. General availability of Oracle Database 21c for on-prem platforms (including Exadata, Linux and Windows) will follow along in 2021.

Practices

Please review [Practices Environment](#) for prerequisite steps you should perform before completing any of the practices.

Note that the practices are designed to be independent from one practice to another. In case a particular configuration is already enabled in your testing database, it is suggested that you re-create your database.

You can also explore the [LiveLabs workshops](#) for Oracle Database 21c which offer hands-on labs on the Oracle Cloud.

Database Features and Licensing App

Use the [Database Features and Licensing app](#) to view feature availability across Oracle Database releases and to see what features are new in Oracle Database 21c.

Database 21c Interactive Architecture Diagram

For more information about the technical architecture of Oracle Database 21c, visit <https://docs.oracle.com/en/database/oracle/oracle-database/21/dbiad/index.html>.

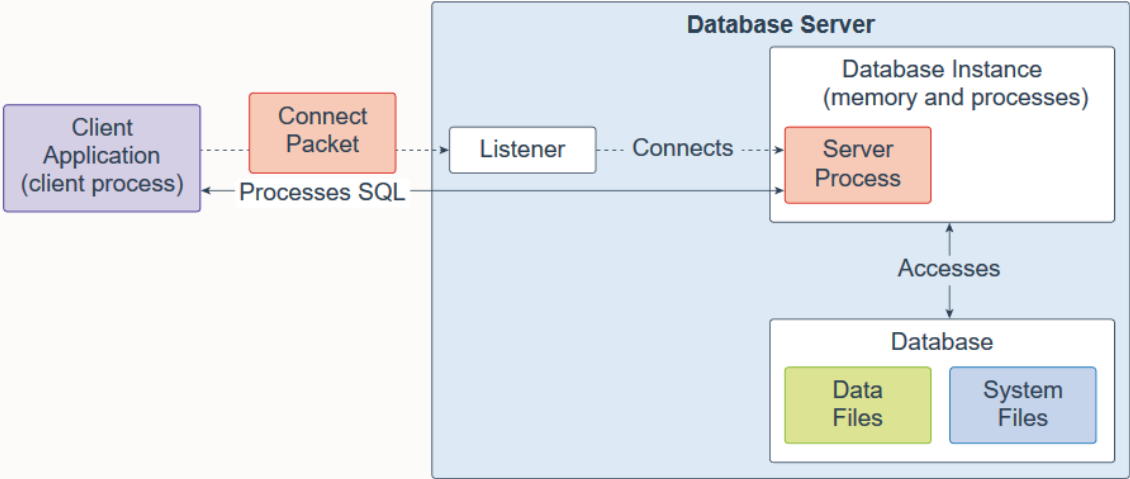
Oracle Database 21c Technical Architecture

This interactive diagram shows the Oracle Database 21c technical architecture.

[Show PDF](#) | [Show Instructions](#)

[First](#) [Previous](#) [Next](#) Slide 1 of 29

Database Server



Practices Environment

Overview

The practices are designed to be independent of each other. However, before starting a practice it is suggested that you re-create your database in your testing environment.

You should not run these labs on a production system. Commands and scripts are provided solely for testing purposes and need some adaptation to your testing environment.

Step 1 : Create an Oracle Database 21c instance

- Create an Oracle Database 21c instance. If you plan to create the instance in Oracle Cloud Infrastructure, follow the instructions described in the [Create an Oracle Cloud Infrastructure VM Database](#) tutorial.
- In the tutorial, in Lab 4 *Create an Oracle Cloud Infrastructure VM Database - Step 2 Create a Database Virtual Machine* - Substep 4 define the CDB and PDB as:
 - **Database name:** Choose default database name to "cdb21".
 - **PDB name** field, enter "pdb21".

Step 2 : Define and test the connections

- Log in to the VM of your Database Virtual Machine. Following the tutorial, Lab 4 *Create an Oracle Cloud Infrastructure VM Database - Step 3 Gather system details and connect to the Database using SSH* to get the IP address of the node and the procedure to log in to the VM.
- Create the net service name alias CDB21 for the container database CDB21 and the net service name alias PDB21 for the pluggable database PDB21.
 - Run the `lsnrctl` utility to find the path of the `/u01/app/oracle/homes/OraDB21Home1/network/admin/tnsnames.ora` file. The letters in the Unix prompt following the login username (opc or root or oracle) are the Hostname prefix defined during the Database VM creation via the tutorial, Lab 4 *Create an Oracle Cloud Infrastructure VM Database - Step 2 Create a Database Virtual Machine - Substep 3 On the DB System Information form*.


```
[oracle@xx ~]$ lsnrctl status

LSNRCTL for Linux: Version 21.0.0.0.0 - Production on 11-DEC-2020 09:28:09

Copyright (c) 1991, 2020, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=xxxxx.sub12100925130.vcndj.oraclevcn.com) (PORT=1521)))
STATUS of the LISTENER
-----
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 21.0.0.0.0 - Production
Start Date                10-DEC-2020 09:43:06
Uptime                    0 days 23 hr. 45 min. 3 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/homes/OraDB21Home1/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/xxxxx/listener/alert/log.xml
Listening Endpoints Summary...
...
Service "CDB21_fra1xn.sub12100925130.xxxxx.oraclevcn.com" has 1 instance(s).
  Instance "CDB21", status READY, has 1 handler(s) for this service...
Service "pdb21.sub12100925130.vcndj.oraclevcn.com" has 1 instance(s).
  Instance "CDB21", status READY, has 1 handler(s) for this service...
```

The OraDB21Home1 sub-directory is the sub-directory mentioned in the /u01/app/oracle/inventory/ContentsXML/inventory.xml file.



```
<HOME_LIST>
<HOME_NAME="OraDB21Home1" LOC="/u01/app/oracle/product/21.0.0/dbhome_1" TYPE="O" IDX="1">
</HOME_LIST>
```

- Open the /u01/app/oracle/homes/OraDB21Home1/network/admin/tnsnames.ora file. There is a net service alias for the container database, CDB21, using the databaseUniqueName. Replace the SERVICE_NAME entry with the value found in the service from lsnrctl, such as CDB21_fra1xn.sub12100925130.xxxxx.oraclevcn.com.

```
[oracle@xx ~]$ vi /u01/app/oracle/homes/OraDB21Home1/network/admin/tnsnames.ora

# tnsnames.ora Network Configuration File: /u01/app/oracle/homes/OraDB21Home1/network/admin/tnsn
# Generated by Oracle configuration tools.

CDB21_FRA1XN =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)
```

- Create an alias entry by copying the CDB alias entry. Replace the CDB alias name with CDB21.

```
CDB21_FRA1XN =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)

CDB21 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)
```

- Create an alias entry for the PDB by copying the CDB alias entry. Replace the CDB alias name with PDB21.

```

CDB21_FRA1XN =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.
com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)

CDB21 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.
com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)

PDB21 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = XX.sub12100925130.xxxx.oraclevcn.
com) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = PDB21_fra1xn.sub12100925130.xxxx.oraclevcn.com)
    )
  )
)

```

- Save the updates in the /u01/app/oracle/homes/OraDB21Home1/network/admin/tnsnames.ora file.
- Test the connection to CDB21.
 - Connect to CDB21 with SQL*Plus.

```

[oracle@xx ~]$ sqlplus sys@CDB21 AS SYSDBA

Enter password: password_defined_during_DBSystem_creation

Connected to:
...
SQL>

```

- Verify that the container name is CDB\$ROOT.

```
SQL> SHOW CON_NAME
```

```
CON_NAME
```

```
-----
```

```
CDB$ROOT
```

```
SQL>
```

- Test the connection to PDB21, and then exit SQL*Plus.
 - Connect to PDB21.

```
SQL> CONNECT sys@PDB21 AS SYSDBA
```

```
Enter password: password_defined_during_DBSystem_creation
```

```
Connected.
```

```
SQL>
```

- Show the container name.

```
SQL> SHOW CON_NAME
```

```
CON_NAME
```

```
-----
```

```
PDB21
```

```
SQL>
```

- Exit SQL *Plus.

```
SQL> EXIT
```

```
[oracle@xx ~]$
```

Step 3 : Download the practices scripts

- Download the [Cloud_21c_labs_for_practices.zip](#) file to the home directory of the login user opc, /home/opc on your VM.
- Copy the /home/opc/Cloud_21c_labs_for_practices.zip to /home/oracle/Cloud_21c_labs_for_practices.zip.

```
[opc@xx ~]$ sudo su -
Last login: Thu Dec 10 09:49:37 UTC 2020
[root@xx ~]# cp /home/opc/Cloud_21c_labs_for_practices.zip /home/oracle/
[root@xx ~]# chown oracle:oinstall /home/oracle/Cloud_21c_labs_for_practices
.zip
[root@xx ~]# exit
logout
[opc@xx ~]$
```

- Unzip Cloud_21c_labs.zip.

```
[opc@xx ~]$ sudo su - oracle
[oracle@xx ~]$ unzip /home/oracle/Cloud_21c_labs_for_practices.zip
Archive: Cloud_21c_labs_for_practices.zip
  creating: labs/
  creating: labs/M104786GC10/
  inflating: labs/M104786GC10/startup.sql
  inflating: labs/M104786GC10/open_keystore_set_key.sql
  inflating: labs/M104786GC10/hr_cre.sql
...
[oracle@xx ~]$
```

Step 4 : Update the practices scripts to the current environment

- Execute 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.
 - Make the script readable, writable, and executable by everyone.

```
[oracle@xx ~]$ sed -i -e "s/\r//g" /home/oracle/labs/update_pass.sh
[oracle@xx ~]$ sed -i -e "s/\n//g" /home/oracle/labs/update_pass.sh
[oracle@xx ~]$ chmod 777 /home/oracle/labs/update_pass.sh
[oracle@xx ~]$
```

- Run the script.

```
[oracle@xx ~]$ /home/oracle/labs/update_pass.sh
Enter the password you set during the DBSystem creation: password_define
d_during_DBSystem_creation
[oracle@xx ~]$
```

Application Development

- [Application Express](#)
- [Globalization](#)
- [Java in Oracle Database](#)
- [JDBC](#)
- [JSON Document Store](#)
- [Net Services](#)
- [Oracle Call Interface](#)
- [Oracle Data Provider for .NET \(ODP.NET\)](#)
- [PL/SQL](#)
- [SQL](#)
- [Text](#)

Application Express

- Application Backups
- Dark Mode
- Export App as Zip
- Faceted Search
- Friendly URL Syntax
- Mega Menus
- Native PDF Printing
- New Data Upload
- New Form Region
- New Team Development
- Oracle JET Charts
- Popup List of Values (LOV)
- Redwood UI
- Remote Application Deployment
- REST Read / Write Enhancements
- Shared List of Values (LOV)
- SQL Workshop Now Supports Simple Oracle Document Access (SODA)
- Universal Theme Enhancements

Application Backups

Oracle Application Express automatically backs up modified applications as a part of daily maintenance.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Dark Mode

Decrease eye strain by enabling Dark Mode.

The development environment can now render with a darker color scheme, which reduces eye strain and is especially helpful for developing late into the night.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Export App as Zip

You can now export your Application Express apps as a single zip file, which can facilitate integration with GIT / SVN and simplify incremental deployment. The zip file mirrors the directory structure of the APEXExport -split utility and splits your application, pages, and shared components into separate files for easy management.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Faceted Search

Faceted search, or faceted navigation, provides an easy-to-use interface to narrow the results of a report and quickly locate specific data.

Like other declarative data-driven components in Application Express, you can create a faceted search page on an existing table, an arbitrary SQL query, an ORDS REST-Enabled SQL Service, or virtually any REST endpoint. Using an intelligent auto-discovery and scoring algorithm, recommended facets are automatically created based upon data distribution, related tables, column sizes, distinct values, average data length, and more. You can customize the recommended facets and easily add new facets using a robust set of declarative attributes.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Friendly URL Syntax

Oracle Application Express application URL syntax has been simplified to support friendlier URLs at runtime. Friendly URLs features application or page numbers, and instead, uses the workspace path prefix, application and page aliases, and standard web parameter syntax for its URL structure.

The new syntax provides a Search Engine Optimization (SEO)-friendly URL structure which is far easier to understand and provides immediate context as to where you are within an app.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Mega Menus

The Universal Theme now supports Mega Menu navigation. Mega Menu navigation renders app navigation as a collapsible floating panel that displays all navigation items at once. Mega Menu navigation is especially useful for displaying the various aspects of your app and enables direct access to app sections.

Mega Menu navigation includes a number of template options that enable you to customize the layout and appearance so it uniquely works for your app. You can extend Mega Menu items to include icons, descriptions, and even badges using custom list attributes. No matter the layout for larger screens, the mega menu gracefully adjusts its appearance to fit small screen devices. Mega Menus are fully accessible and keyboard friendly.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Native PDF Printing

You can now print PDF files directly from interactive grids. This feature produces a PDF file which includes formatting options such as highlighting, column grouping, and column breaks. To make use of this feature, simply enable PDF as an additional download format in your region attributes.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

New Data Upload

The data upload functionality has been modernized to support native Excel, CSV, XML and JSON documents.

Data upload in SQL Workshop features a new drag and drop user interface that provides support for uploading native Excel, CSV, XML and JSON documents into a new table or into existing tables. These same capabilities can be accessed from the Create Application Wizard by selecting the From a File option.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

New Form Region

A new Form region type provides superior functionality compared to the old legacy form pages.

A new Form region type provides support for internal and external data sources (that is, Local Database, REST Enabled SQL Service, or Web Source). Forms can be created on SQL queries, or by creating PL/SQL Functions returning SQL Queries. In addition, this new Form region type supports more than two Primary Key Columns and includes more control on Lost Update detection and Row Locking. Finally, you can migrate existing legacy form pages to the new form region with the Upgrade Application function in App Builder Utilities.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

New Team Development

The Team Development application has been completely reimagined to provide a simple and easy way for your team to collaborate together.

Team Development provides a simple, conversational approach to team communication. Create an issue to track everything in a single place. You can classify issues using label groups and labels, create templates to provide users with starter text for issues and comments, and track progress by assigning milestones. Individual users can register their interest in specific issues and define at a general level what attributes they are interested in watching. Users can also choose whether they wish to be notified in the application, by email, or both.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Oracle JET Charts

With every release, Oracle Application Express upgrades Oracle JET Charts. Oracle Application Express release 20.1 includes JavaScript Extension Toolkit (JET) 8.3.0.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Popup List of Values (LOV)

Oracle Application Express introduces the completely redesigned Popup List of Values (LOV) which now supports multiple display columns, multiple return values, search-as-you-type, improved multiple value selection, and an all-new simplified user interface.

Redwood UI

The Oracle Application Express user interface has been refreshed to align better with Redwood, Oracle's new user experience design language.

The new design and color scheme extends across the full developer experience and provides refreshing new visuals. The appearance of Application Express can now automatically switch between the dark and light appearance modes based on your OS or platform setting.

Remote Application Deployment

Deploy an application to remote Oracle Application Express instances using REST Enabled SQL references.

One-click remote application deployment leverages the existing REST Enabled SQL references that works with a Oracle REST Data Services (ORDS) REST Enabled SQL Service.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

REST Read / Write Enhancements

Read and write data using REST Enabled SQL references or Web Source Modules.

Define a REST Enabled SQL reference once and use it across a workspace. Create Web Source Modules to define the source within each application. Both approaches are fully declarative, using Data Discovery with parameters. Forms and interactive grids now include built-in support for the internal or external data source you specify.

Related Topics

- [Oracle® Application Express SQL Workshop Guide](#)

Shared List of Values (LOV)

Shared List of Values (LOV) in Oracle Application Express include significant enhancements and new functionality. Shared LOVs now support the full set of data sources (table, SQL Query, function returning SQL query, ORDS REST-Enabled SQL Service and Web Source Modules), declarative column mappings, multiple display columns, grouping, embedded icons, and more.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

SQL Workshop Now Supports Simple Oracle Document Access (SODA)

SQL Workshop has now been extended to support Simple Oracle Document Access (SODA) collections. Create and store collections of documents in Oracle Database, retrieve them, and query them, without needing to know Structured Query Language (SQL) or how the documents are stored in the database. You can load data to a SODA collection as JSON, TXT, or ZIP files.

Related Topics

- [Oracle® Application Express SQL Workshop Guide](#)

Universal Theme Enhancements

The Universal Theme includes UI refinements, accessibility improvements, new template options, and Theme Roller enhancements.

Universal Theme contains many accessibility improvements including a new Skip to Main Content link. The Universal Theme also includes two additional styles for the tree-based Navigation Menu and a new Inline Popup region template. Finally, a number of components and styles have been visually tweaked and refined to provide a more streamlined user interface.

Related Topics

- [Oracle® Application Express App Builder User's Guide](#)

Globalization

- [Enhanced Parallel Execution Applicability](#)
- [New German Linguistic Sorts for Capital Sharp S Support](#)
- [New Era Support for Japanese Imperial Calendar](#)
- [Unicode 12.1 Support](#)
- [Zero Downtime Upgrade of Timezone Data](#)

Enhanced Parallel Execution Applicability

Parallel execution can now be used for abstract data types, character keys with linguistic sorts, and timestamp with time zone.

Broadening the applicability of parallel execution to non-native data types can increase the performance of applications using these data types.

New German Linguistic Sorts for Capital Sharp S Support

Two new linguistic sorts (XGERMAN_S and XGERMAN_DIN_S) are added to Oracle Database to support Latin Capital Letter Sharp S as the uppercase form of Latin Smaller Letter Sharp S.

This feature expands the database linguistic support to meet the requirements of the German market.

Related Topics

- [Oracle® Database Globalization Support Guide](#)

New Era Support for Japanese Imperial Calendar

The new Japanese era Reiwa, which went into effect on May 1, 2019, is now supported in Oracle Database for the Japanese Imperial Calendar.

The Japanese Imperial Calendar is used widely in Japan in addition to the Gregorian Calendar. Supporting the new era is an essential requirement for software distributed in the Japanese market.

Related Topics

- [Oracle® Database Globalization Support Guide](#)

Unicode 12.1 Support

The National Language Support (NLS) data files for AL32UTF8 and AL16UTF16 character sets are updated to match version 12.1 of the Unicode Standard character database. The support for Unicode Collation Algorithm (UCA) is also updated to conform with UCA 12.1.

This enhancement enables Oracle Database to conform to the latest version of the Unicode Standard.

Related Topics

- [Oracle® Database Globalization Support Guide](#)

Zero Downtime Upgrade of Timezone Data

The process of upgrading timezone data to reflect up-to-date Governmental Daylight Saving Time rules and to change existing data to reflect these new rules is accomplished in a streamlined, simplified, and automated manner without the need of any downtime or significant impact on the availability of timezone data for queries and DML operations.

Zero downtime upgrade of timezone information removes the burden of database administrators to schedule a downtime window for this operation, removing complexity and obstacles towards continuous 24x7 availability of any database system.

Related Topics

- [Oracle® Database Database Globalization Support Guide](#)

Java in Oracle Database

- [Oracle JVM Security Enhancements](#)

Oracle JVM Security Enhancements

Oracle Java virtual machine (JVM) security has been enhanced to make it well suited for pluggable databases (PDBs). The enhancements also include support for the new Java module system and safeguarding of Oracle JVM against security vulnerabilities.

The enhanced Oracle JVM security makes it safe to use Oracle JVM with pluggable databases (PDB) and shields it from security vulnerabilities like the Spectre attack. In addition, it simplifies developing Java in the database applications using the new module system.

Related Topics

- [Oracle® Database Java Developer's Guide](#)

JDBC

- [Java Library for Reactive Streams Ingestion](#)
- [JDBC DataSource for Sharded Databases Access](#)
- [JDBC Reactive Extensions](#)
- [JDBC Support for Native JSON Data Type](#)

Java Library for Reactive Streams Ingestion

This new Java library allows high-speed ingestion of data streams with non-blocking back pressure.

Java applications that use the provided APIs may continuously receive and ingest data from a large group of clients.

The ingestion process is non-blocking and extremely fast through the direct path load into the database tables. The library performs affinity grouping by destination including Oracle Real Application Clusters (RAC) nodes, Oracle Active Data Guard, and Sharded database. The library performs CPU optimization by decoupling record processing from database IO. Through the Universal Connection Pool (UCP), the ingestion process furnishes high availability and scalability.

This feature enables implementing high-speed ingestion of streaming data including sensors data, time series (trading), Call Detail Records (CDRs), geoSpatial activities, social media feeds, web site logs, and so on, with scalability and high availability.

Related Topics

- [Oracle® Database JDBC Developer's Guide](#)

JDBC DataSource for Sharded Databases Access

You can use the new JDBC datasource to simplify the access of Java applications to sharded databases, where the applications can connect to the sharded databases transparently, with almost no change to the application code.

This feature enables Java connectivity to a sharded database without the need for a Java application to furnish a sharding key because the sharding data source derives the sharding key from the SQL statement. Your Java applications can scale out to sharded databases transparently as there is little to no change to the application code.

The sharding data source optimizes the performance of your applications as you do not need to configure the Universal Connection Pool (UCP) or create separate connection pools for cross-shard statements and single-shard statements. Also, you do not have to check-in or check-out a physical connection for every new sharding key because the sharding data source does it automatically.

Related Topics

- [Oracle® Database JDBC Developer's Guide](#)

JDBC Reactive Extensions

The Reactive Extensions are a set of methods that extend the JDBC standard to offer asynchronous database access with back pressure support. The Reactive Extensions implement the Publisher and Subscriber types defined by `java.util.concurrent.Flow`. Flow is the JDK's standard representation of a reactive stream.

The Reactive Extensions use non-blocking mechanisms for creating Connections, executing SQL, fetching rows, committing, rolling back, closing Connections, and reading and writing BFILEs, BLOBs, and CLOBs.

The extensions bring scalability, high throughput, and Reactive Streams support to Java applications that use the Oracle Database in the Cloud and on-premises. This extension works as a Service Provider for the Reactive Streams libraries including: Reactor, RxJava, and Akka Streams.

Related Topics

- [Oracle® Database JDBC Developer's Guide](#)

JDBC Support for Native JSON Data Type

This release introduces a new JSON datatype for efficient storage of JSON data. The package `oracle.sql.json` provides classes and interfaces for working with SQL JSON type values.

This feature furnishes a simpler and richer type system (that is, support for dates, timestamps), no constraint check (IS JSON) and improves the performance of Java applications (that is, faster access to nested JSON values). A JSON parser inside the JDBC driver 21c allows optimizations during the conversion from Java String to native JSON and the other way around.

Related Topics

- [Oracle® Database JDBC Java API Reference](#)

JSON Document Store

- New JSON Data Type
- New Oracle SQL Function JSON_TRANSFORM
- SQL/JSON Syntax Improvements
- Multivalue Index for JSON
- Enhancements to View Creation and of Virtual Columns Addition
- JSON Scalar Allowed at Top Level of JSON Document (RFC 8259 Support)
- Compatibility for Old Clients With Binary JSON

New JSON Data Type

JSON is a new SQL and PL/SQL data type for JSON data. It provides a substantial increase in query and update performance.

The JSON data type is optimized for query and DML processing. It can yield significant database performance improvements for processing JSON data.

You can use the JSON data type in most places where a SQL data type is allowed, including:

- As the column type for table or view DDL
- With SQL/JSON functions and conditions, and with PL/SQL procedures and functions
- In Oracle dot-notation query syntax
- For creation of functional and search indexes

Oracle Call Interface (OCI) and Java Database Connectivity (JDBC) clients now provide APIs that can work directly with binary JSON data, significantly saving network costs and server CPU cycles.

Database In-Memory supports the ability to populate JSON data type columns in the In-Memory column store. JSON data defined with the JSON data type is populated in a proprietary binary format that is optimized for query processing and can significantly improve JSON processing performance.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

New Oracle SQL Function JSON_TRANSFORM

You can use SQL function `JSON_TRANSFORM` to update parts of a JSON document. You specify which parts to modify, the modifications, and any new values.

`JSON_TRANSFORM` makes it easier for an application to modify a JSON document, without having to parse and rebuild it. In most cases, it also avoids a round-trip between the server and client for the whole document.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

SQL/JSON Syntax Improvements

You can now express more complex SQL/JSON queries and express some queries more succinctly.

- New SQL function `JSON_SCALAR` accepts a scalar instance of a SQL data type and returns a scalar JSON value as an instance of JSON data type.
- New JSON path-language item methods support `JSON_SCALAR: float()`, `double()`, `binary()`, `ymInterval()`, and `dsInterval()`.
- The JSON path-language and dot-notation syntax support new, aggregate item methods: `avg()`, `count()`, `minNumber()`, `maxNumber()`, `minString()`, `maxString()`, and `sum()`.
- You can now express more complex SQL/JSON queries and express some queries more succinctly, and SQL/JSON path-expression syntax for array steps is improved.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

Multivalue Index for JSON

A new create index syntax `CREATE MULTIVALUE INDEX` allows you to create a functional index on arrays of strings or numbers within a JSON type column. Each unique value within the array will become a searchable index entry.

This avoids the need for full JSON scans to find values within arrays in JSON columns, when searched using the `JSON_EXISTS` or `JSON_QUERY` operators. It provides similar benefits to conventional functional indexes when searching JSON, but conventional functional indexes are limited to a single indexed value per row.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

Enhancements to View Creation and of Virtual Columns Addition

`DBMS_JSON.CREATE_VIEW` now gives you the option to create a materialized view instead of a standard view. It also gives you the option to specify a particular path so the view can be created on a subset of the data. Both `CREATE_VIEW` and `ADD_VIRTUAL_COLUMN` are enhanced to allow automatic resolution of column naming conflicts, to provide a prefix to be applied to column names, and to specify the case-sensitivity of column names.

Enhances development flexibility and allows for materialized views, which may improve query performance with a trade-off against DML performance.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

JSON Scalar Allowed at Top Level of JSON Document (RFC 8259 Support)

JSON documents in Oracle Database can now have a top-level JSON scalar value. Previously they had to have a JSON object or array value.

This feature helps Oracle JSON support be compliant with RFC 8259.

This feature will only be available when the database initialization parameter `compatible` is set to 20 or higher.

If the parameter value is at least 20, then JSON data that is stored either textually (`VARCHAR2`, `CLOB`, `BLOB`) or as JSON data type respects RFC 8259: it allows top-level scalars in documents. For a JSON column you can, however, use an `IS JSON` check constraint with keywords `DISALLOW SCALARS` to disallow documents having top-level scalar value.

Related Topics

- [Oracle® Database JSON Developer's Guide](#)

Compatibility for Old Clients With Binary JSON

The database supports JSON in a binary format. In some situations, clients will work directly with the binary format. However, this requires that the client be a recent enough version to understand the format. The database will now identify older clients and automatically transform JSON data from binary to text (serialized) format before transmitting it to them, and convert back from text to binary format as required.

Support for older clients without being forced to upgrade.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

Net Services

- Migration of Oracle CMAN Sessions with High Availability
- Oracle CMAN Traffic Director Mode Support for All Types of Database Links
- REST APIs for Oracle CMAN Administration, Proxy Protocol, Enhanced Rule List, and Bandwidth Management
- Reverse Connection Support Using CMAN Tunnels

Migration of Oracle CMAN Sessions with High Availability

Client/server sessions can be migrated from one Oracle Connection Manager (CMAN) instance to another Oracle CMAN instance during a planned upgrade or while patching Oracle CMAN. Live sessions can be migrated with data in-transit.

Operations that are running either on a client or on a server continue to run seamlessly during the migration with zero downtime. You can also add new client connections during the migration.

Related Topics

- [Oracle® Database Net Services Administrator's Guide](#)

Oracle CMAN Traffic Director Mode Support for All Types of Database Links

Starting with this release, Oracle Connection Manager (CMAN) Traffic Director Mode is extended to support all types of dedicated database links including Fixed User, Connected User, and Current User.

Oracle Connection Manager in Traffic Director Mode enhances application scalability, performance, security, tenant isolation, and high availability (zero downtime during planned and unplanned database outages).

Related Topics

- [Oracle® Database Net Services Administrator's Guide](#)

REST APIs for Oracle CMAN Administration, Proxy Protocol, Enhanced Rule List, and Bandwidth Management

You can use REST APIs to manage Oracle Connection Manager (Oracle CMAN) instances. Proxy protocol provides additional security and access control. Enhanced rule list allows scalability with segregation of rule_lists for each service. You can also manage distribution of bandwidth across services using Oracle CMAN.

This feature offers REST API management, security with proxy protocol, scalability with enhanced rule list, and per service bandwidth management.

Related Topics

- [Oracle® Database Net Services Administrator's Guide](#)

Reverse Connection Support Using CMAN Tunnels

Starting with Oracle Database 21c, you can use secure tunnels to connect to an Oracle Database instance, which is inside a network that supports only outbound connections.

A network may allow only outbound connections and restrict inbound connections for security reasons. However, using the Oracle Connection Manager tunnel feature, you can connect to a database inside a network that allows only outbound connections. Oracle Connection Manager creates a pool of connections, known as tunnels, that can be used to connect to a database inside the network.

[Details: Reverse Connection Support Using CMAN Tunnels](#)

This page provides more detailed information about how to access a database inside a network that allows only egress connections.

Related Topics

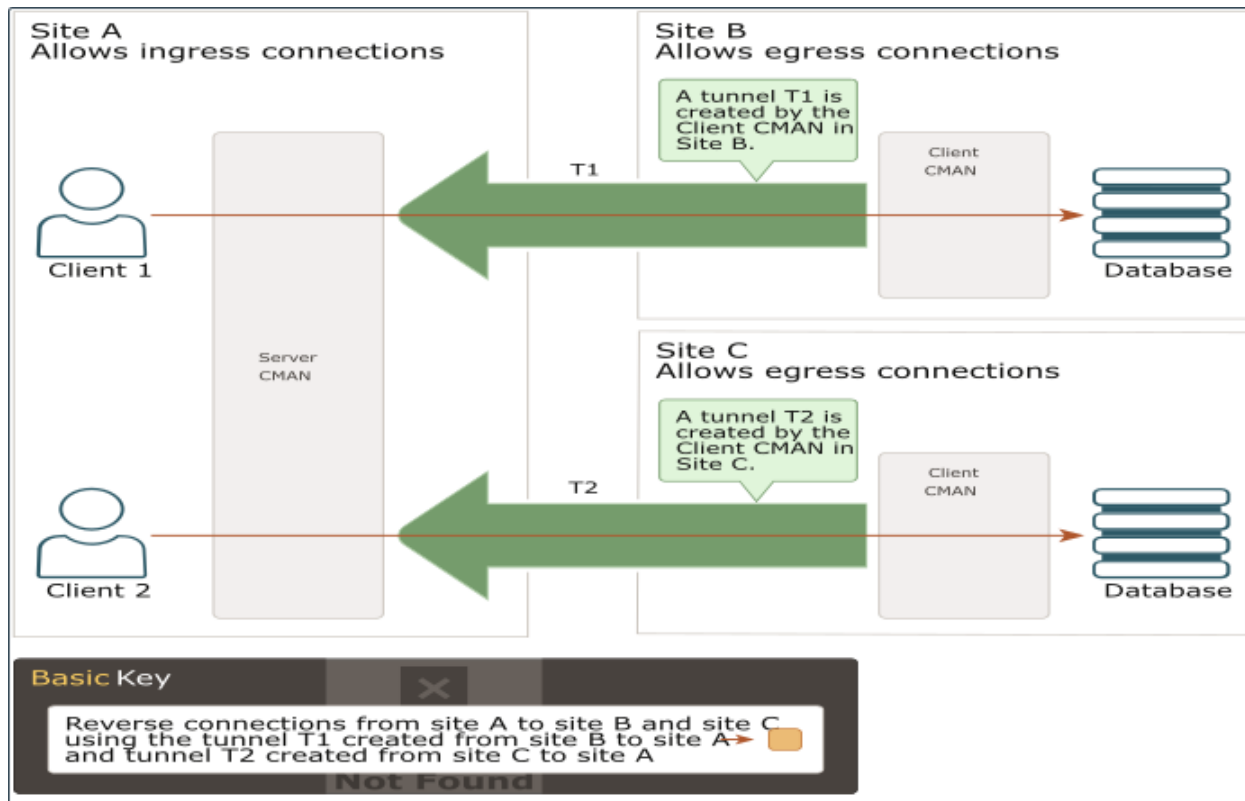
- [Oracle® Database Net Services Administrator's Guide](#)

Details: Reverse Connection Support Using CMAN Tunnels

To access a database inside a network that allows only egress connections, you must deploy CMAN at both the client site that is hosting the database and the server site that wants to access the database. The data transfer happens over an encrypted channel on the public internet using TLS, if TLS is configured between the two CMANs.

When client CMAN is started, the gateway connects to the server CMAN and creates a pool of connections, known as tunnels. Reverse connections from the server to the client are routed through these tunnels. You can also configure the pool size.

In the following figure, the client CMAN uses the tunnel service of the server CMAN to establish a tunnel connection. Once a client CMAN establishes a tunnel, the server CMAN offers the client CMAN identifier as a service for clients in site A.



Oracle Call Interface

- Improvements to OCI Data Interface for LOBs
- New C Client Interface APIs for JSON Data Type
- Oracle Call Interface API to build and quote TNS Connection Strings
- Oracle Call Interface Session Pool Improvements
- Oracle SODA for C and PL/SQL APIs Enhancements
- Support for C99 Syntax in Pro*C/C++ Precompiler

Improvements to OCI Data Interface for LOBs

This feature makes the OCI Data Interface for LOBs more user friendly.

The OCI Data Interface for LOBs is efficient because it reduces the number of round trips between the client and the database. This feature is further improved by removing the need for NULL callbacks and usage of 'ind' variable when the LOB length exceeds the sb2 size.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

New C Client Interface APIs for JSON Data Type

This feature introduces new Oracle Call Interface (OCI) APIs to operate with JSON descriptor. You can read and write textual JSON or binary JSON (in Oracle binary format) from/to a buffer or a stream. You can bind and define a JSON descriptor to write and read JSON data from the database.

With the new JSON APIs, you can now preserve abstraction without a need to serialize the data to text.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

Oracle Call Interface API to build and quote TNS Connection Strings

A new Oracle Call Interface API can now be used to build and quote Oracle Database Transparent Network Substrate (TNS) connection strings.

The Oracle Call Interface enhancement enables validation of values, which you can use for Oracle Net Service connection string attributes.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

Oracle Call Interface Session Pool Improvements

The Oracle Call Interface Session Pool has been enhanced.

Monitoring and administration has been improved.

When the number of sessions exceeds the minimum pool size, idle session cleanup now occurs even if there is no pool activity.

There is better OCISessionGet wait timeout accuracy.

When using Oracle Sharding, the pool has better balancing of session across shards.

You can now manage Oracle Call Interface Session Pool better with performance, auto-tuning, and administration improvements.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

Oracle SODA for C and PL/SQL APIs Enhancements

This feature introduces new SODA APIs for saving documents to a collection, truncating a document collection, and specifying the array fetch size to read documents from a collection, which reduces network round-trips.

The enhancements improve interoperability of the SODA APIs with the database and utilities.

Related Topics

- [Oracle® Call Interface Programmer's Guide](#)

Support for C99 Syntax in Pro*C/C++ Precompiler

The Pro*C/C++ Precompiler now supports the C99 standard, the ISO/IEC 9899:1999 standards specification for C programming.

C99 syntax and semantics allow application developers to use richer functionality.

Related Topics

- [Pro*C/C++ Programmer's Guide](#)

Oracle Data Provider for .NET (ODP.NET)

- [Managed ODP.NET and ODP.NET Core: Bulk Copy](#)
- [Managed ODP.NET and ODP.NET Core: Debug Tracing Redaction](#)
- [Managed ODP.NET and ODP.NET Core: WebSocket and WebSocket Secure](#)

Managed ODP.NET and ODP.NET Core: Bulk Copy

ODP.NET Bulk Copy enables applications to efficiently load large amounts of data from a table in one database to another table in a different database. Managed ODP.NET and ODP.NET Core now support Bulk Copy and all its APIs.

ODP.NET Bulk Copy is the most optimized .NET solution when a large data set needs to be loaded into a table or between database tables in different databases.

Managed ODP.NET and ODP.NET Core: Debug Tracing Redaction

Managed ODP.NET and ODP.NET Core has introduced a new trace level, that can exclude SQL statements and network packet contents from being included in the trace file.

Managed ODP.NET and ODP.NET Core: WebSocket and WebSocket Secure

Managed ODP.NET and ODP.NET Core now support both WebSocket and WebSocket Secure. Unmanaged ODP.NET already supports these protocols.

WebSocket has the ability to engage in two-way communication simultaneously using a single connection between web application and host unlike HyperText Transfer Protocol (HTTP). WebSocket Secure uses SSL/TLS to encrypt communications.

PL/SQL

- [DBMS_CLOUD Package](#)
- [JavaScript Execution using DBMS_MLE](#)
- [New PL/SQL Iterator Constructs](#)
- [New Pragma SUPPRESSES_WARNING_6009](#)
- [PL/SQL Qualified Expressions Enhancements](#)
- [PL/SQL Support For New JSON SQL Data Type](#)
- [PL/SQL Type Attributes in User-Defined Types](#)

DBMS_CLOUD Package

Oracle provides two core mechanisms to work with data in object stores, as part of the new `DBMS_CLOUD` package or manually defining external tables.

Using `DBMS_CLOUD` provides benefits and additional functionality that goes beyond DDL and is fully compatible with Oracle Autonomous Database. Oracle strongly recommends leveraging the new `DBMS_CLOUD` package over manual external table creation.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

JavaScript Execution using DBMS_MLE

The `DBMS_MLE` package allows users to execute JavaScript code inside the Oracle Database and exchange data seamlessly between PL/SQL and JavaScript. The JavaScript code itself can execute PL/SQL and SQL through built-in JavaScript modules. JavaScript data types are automatically mapped to Oracle Database data types and vice versa.

With the `DBMS_MLE` package, developers can write their data processing logic in JavaScript. JavaScript is a widely-used and popular programming language that can now also be used for writing programs that need to execute close to the data.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

New PL/SQL Iterator Constructs

PL/SQL is enhanced to help you program iteration controls using new iterators in loops and in qualified expressions.

The new iterator constructs are clear, simple, understandable, and efficient.

Related Topics

- [Oracle® Database PL/SQL Language Reference](#)

New Pragma SUPPRESSES_WARNING_6009

The `SUPPRESSES_WARNING_6009` pragma allows more robust error handling and better encapsulation and modularization.

The PL/SQL compiler issues warning `PLW-06009` if it determines that an `OTHERS` exception handler does not, in all cases, end in either an explicit `RAISE` statement or in a call to the PL/SQL supplied procedure `RAISE_APPLICATION_ERROR`. The compiler's behavior is too aggressive for some programming styles when programmers supply their own reporting subroutines. This new pragma quiets the warning.

Related Topics

- [Oracle® Database PL/SQL Language Reference](#)

PL/SQL Qualified Expressions Enhancements

Starting with this release, three new types of iterator choice association are added for use in qualified expressions. The basic iterator choice association extends the current iterator choice association by allowing a full iterator as the index. The index iterator choice association provides an index expression along with the value expression. The sequence iterator choice association allows a sequence of values to be added to the end of a collection. In each case, the expressions specified may reference the iterands.

Qualified expressions improve program clarity and programmer productivity.

Related Topics

- [Oracle® Database PL/SQL Language Reference](#)

PL/SQL Support For New JSON SQL Data Type

You can use the new JSON SQL data type in PL/SQL.

The new JSON data type allows you to pass JSON data from SQL to PL/SQL and back to SQL (static and dynamic). PL/SQL now supports binding directly JSON data from client side interfaces, such as Oracle Call Interface (OCI) and Java Database Connectivity (JDBC), as well, as from PL/SQL to callouts. JSON can be a differentiating type in overload resolution.

Related Topics

- [Oracle® Database PL/SQL Language Reference](#)

PL/SQL Type Attributes in User-Defined Types

You can use attributes of PL/SQL scalar data types, such as `BOOLEAN` and `PLS_INTEGER`, in non-persistable object types.

You can use non-persistable object types in your PL/SQL code if you have no desire to persist instances of these types. This is useful when you are developing programs following Oracle's object oriented programming model.

Related Topics

- [Oracle® Database PL/SQL Language Reference](#)

SQL

- [Enhanced SQL Set Operators](#)
- [Expression Support for Initialization Parameters](#)
- [Placeholders in SQL DDL Statements](#)
- [SQL Macros](#)

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](#)

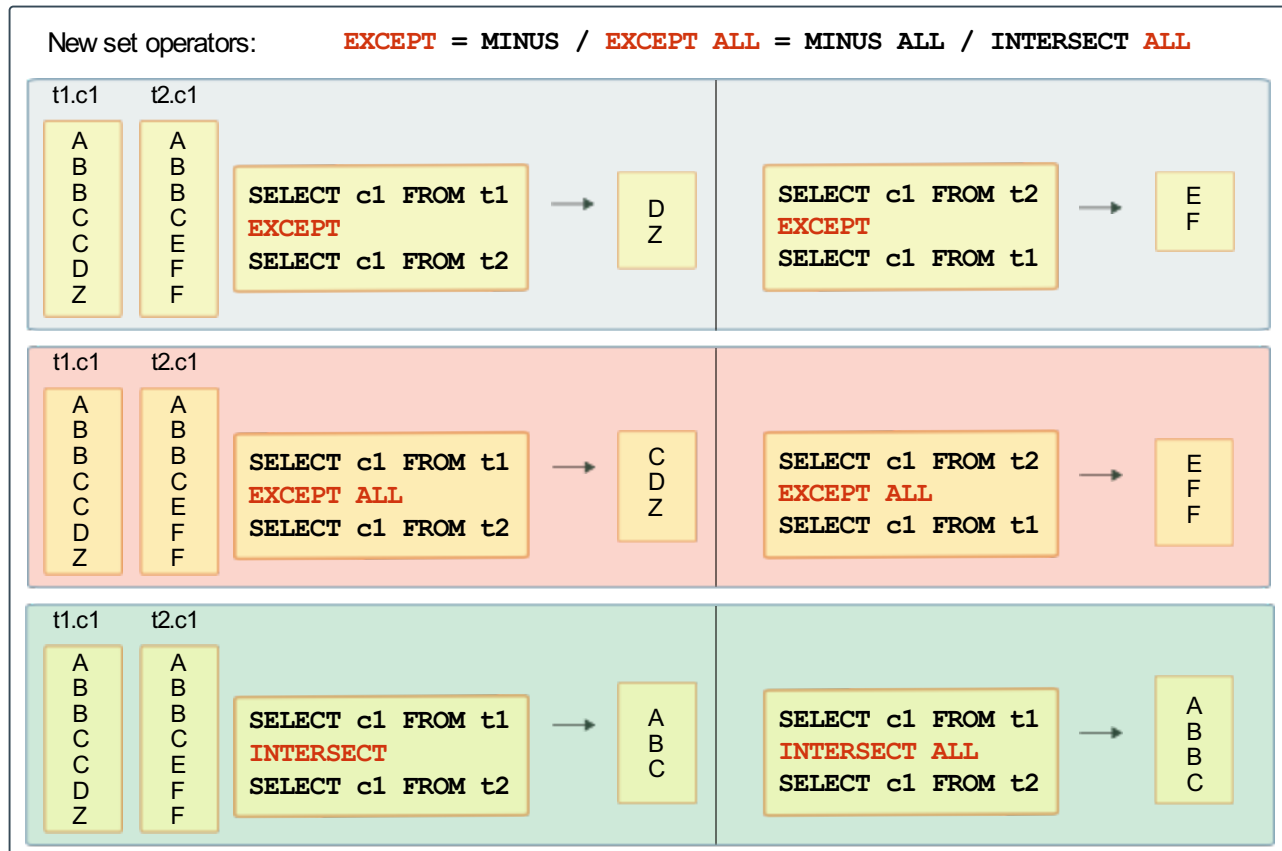
This practice shows how to use the new set operators, `EXCEPT`, `EXCEPT ALL` and `INTERSECT ALL`.

Related Topics

- [Oracle® Database SQL Language Reference](#)

Details: Enhanced SQL Set Operators

This page provides more detailed information about new SQL set operators and enhancements to existing operators.



Until Oracle Database 21c, only the UNION set operator could be combined with ALL. Oracle Database 21c introduces two set operators, MINUS ALL (same as EXCEPT ALL) and INTERSECT ALL.

In the examples in the graphic, the first and second statements combine results from two queries with the EXCEPT operator (being equivalent to MINUS) and return only unique rows returned by the first query but not by the second query.

The third and fourth statements combine results from two queries with the EXCEPT ALL operator (being equivalent to MINUS ALL) and return only rows returned by the first query but not by the second query, even if not unique.

The fifth and sixth statements combine results from two queries with the INTERSECT operator and return only unique rows returned by both queries.

Practice: Using New Set Operators

Overview

This practice shows how to use the new set operators, EXCEPT, EXCEPT ALL and INTERSECT ALL.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- 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 21c Enterprise Edition Release 21.0.0.0.0 -
Production
Version 21.2.0.0.0

$
```

Step 2: Test the set operator with the `EXCEPT` clause

- Connect to `PDB21` as `OE`.

```
$ sqlplus oe@PDB21

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

Enter password:
Last Successful login time: Mon Mar 16 2020 11:32:00 +00:00

Connected to:

SQL>
```

- 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>

```

- 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>

```

- 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>

```

Step 3: Test the set operator with the `EXCEPT ALL` clause

- Would the usage of ALL in the set operator defined in a query in a previous step 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. This does

not mean anything relevant. The use of `ALL` in operators must be appropriate.

Step 4 : Test the set operator with the `INTERSECT` clause

- 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>
```

Step 5 : Test the set operator with the `INTERSECT ALL` clause

- Would the usage of `ALL` in the operator defined in the query 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.

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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Test

- Log in to CDB21 as SYSTEM.

```
$ sqlplus system

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

Enter password:
Last Successful login time: Mon Mar 16 2020 08:49:41 +00:00

Connected to:

SQL>
```

- Set `SGA_TARGET` to 2G.

```
SQL> ALTER SYSTEM SET sga_target = 15G;
ALTER SYSTEM SET sga_target = 15G
*
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 `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   13824M
sga_min_size                        big integer   0
sga_target                          big integer   11072M

SQL>
```

- Set `JOB_QUEUE_PROCESSES` to 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	40
log_archive_max_processes	integer	4
processes	integer	400

```
SQL>
```

- Set 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	40
db_writer_processes	integer	1
gcs_server_processes	integer	0
global_txn_processes	integer	1
job_queue_processes	integer	40
log_archive_max_processes	integer	4
processes	integer	400

```
SQL>
```

- What happens if you change the value of processes?
 - Set the processes value to 500 in the SPFILE.

```
SQL> ALTER SYSTEM SET PROCESSES = 500 SCOPE=SPFILE;

System altered.

SQL>
```

- o 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>
```

- o 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 restart.

- Set `db_recovery_file_dest` to the same value as `$HOME` in CDB21.

```
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                          250G
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
./CDB21_FRA1XN:
total 4
drwxr-x--- 3 oracle oinstall 4096 Dec 14 10:21 archivelog

./CDB21_FRA1XN/archivelog:
total 4
drwxr-x--- 2 oracle oinstall 4096 Dec 14 10:21 2020_12_14

./CDB21_FRA1XN/archivelog/2020_12_14:
total 499832
-rw-r----- 1 oracle oinstall 511804416 Dec 14 10:21 o1_mf_1_4_hxgh534q_.arc
-rw-r----- 1 oracle oinstall    11264 Dec 14 10:21 o1_mf_1_6_hxgh575g_.arc
-rw-r----- 1 oracle oinstall    2560 Dec 14 10:21 o1_mf_1_5_hxgh54jb_.arc
...
$ exit
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. `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. `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.

19c Statements not supporting bind values:

- Some statements are subject to SQL Injection attacks unless developers are careful about avoiding SQL Injection attacks by using techniques such as input validation, quoting user input appropriately.

21c Statements supporting placeholder values:

- Placeholders can be added in statements like:

```
CREATE USER :!username IDENTIFIED BY :!password
DEFAULT TABLESPACE example QUOTA 10M ON example
PROFILE app_user PASSWORD EXPIRE;
```

- `OCIStmtPlaceholderSubstitute()` is called to substitute the placeholders strings in SQL statements. Substitution takes place before the statement is executed.
- Placeholders can be specified in only those statements that cannot have bind variables.
- User input strings are either validated or quoted before they are substituted in the SQL text: distinct modes determine the behavior.
- The `OCIStmtPlaceholderSubstitute()` call for the username in the statement could be:

```
OCIStmtPlaceholderSubstitute(stmtHP, "username", strlen("username"),
"scott", strlen("scott"), OCI_DEFAULT);
```

- This mitigates the risk from SQL injection attacks.

The statements that cannot have OCI placeholders are those beginning with keywords such as `SELECT`, `UPDATE`, `DELETE`, `INSERT`, `BEGIN`, `DECLARE`, `RETURNING`, `CALL`, `MERGE`, `ROLLBACK`, `COMMIT`, and `FLASHBACK` because 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 21c](#).

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 also provides more detailed information, explains what SQL Macros are useful for, and supplies various examples to use.

Application developers can use SQM to write macros, useful in SQL statements:

- As **scalar expressions**: typically used in `SELECT`, `WHERE`, and `HAVING` clauses
- As **table expressions**: typically used in a `FROM` clause
- As **pseudo operators**

SQM Scalar Expressions - Example 1

1. Create the SQL macro:

```
SQL> CREATE FUNCTION concat_self(str varchar2, cnt pls_integer)
RETURN VARCHAR2 SQL_MACRO(SCALAR)
IS BEGIN RETURN 'rpad(str, cnt * length(str), str)';
END;
/
```

2. Use the SQL macro in a query in the `SELECT` clause:

```
SQL> SELECT ename, concat_self(ename, :num) FROM emp;
```

-> Rewritten as:

```
SELECT ename, rpad(ename, :num * length(ename), ename
FROM emp;
```

```
DBA_PROCEDURES.SQL_MACRO => NULL, SCALAR, TABLE
```

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`, 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 parameterized views. SQL macros increase developer productivity, simplify collaborative development, and improve code quality.

The example in the slide shows a SQL macro written as a scalar expression used in the `SELECT` list of the query.

SQM Scalar Expressions - Example 2

1. Create the SQL macro:

```
SQL> CREATE FUNCTION clip(lo VARCHAR2, x VARCHAR2, hi VARCHAR2)
RETURN VARCHAR2 SQL_MACRO(SCALAR)
IS BEGIN RETURN 'least(greatest(x, lo), hi)';
END;
/
```

2. Use the SQL macro in a query in the SELECT clause:

```
SQL> SELECT ename, clip(:lower, sal+comm, :upper) FROM emp;
```

-> Rewritten as:

```
SELECT ename, least(greatest(sal+comm, :lower), :upper)
FROM emp;
```

3. Use the SQL macro in a query in both the SELECT and WHERE clauses:

```
SQL> SELECT clip(1000,sal,2000) FROM emp
WHERE clip(SYSDATE-10, hiredate, SYSDATE+10) = hiredate;
```

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.

SQM Table Expressions - Example 3 - Polymorphic Non-parameterized Views

View versus SQM

1. Create a view:

```
CREATE VIEW v_budget AS
SELECT deptno,
sum(sal) v_budget
FROM emp
GROUP BY deptno;
```

2. Query the view:

```
SELECT * FROM v_budget
WHERE deptno = :dno;
```

1. Create an SQM:

```
CREATE FUNCTION mbudget
RETURN VARCHAR2 SQL_MACRO
IS BEGIN
RETURN q'(SELECT deptno, sum(sal) budget
FROM emp GROUP BY deptno)';
END;
```

2. The original query

```
SELECT * FROM mbudget()
WHERE deptno = :dno;
```

gets rewritten as:

```
SELECT * FROM
(SELECT deptno, sum(sal) budget
FROM emp
GROUP BY deptno)
WHERE deptno = :dno;
```

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

Overview

These practices show how to use SQL Macros as scalar and table expressions.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Use SQL Macros as a scalar expression

- Ensure that `PDB21` is opened. If it is not opened, open it.

```
$ sqlplus / AS SYSDBA

Connected.

SQL> SHOW PDBS

  CON_ID  CON_NAME                                OPEN MODE  RESTRICTED
-----  -
         2 PDB$SEED                                READ ONLY  NO
         3 PDB21                                READ WRITE NO

SQL>
```

- Create the `HR` schema and its tables in `PDB21`.

```
SQL> CONNECT sys@pdb21 AS SYSDBA
Enter password:
Connected.
SQL> @$HOME/labs/M104780GC10/hr_main.sql password users temp /home/oracle/lab
s /home/oracle/labs
specify password for HR as parameter 1:

specify default tablesppace for HR as parameter 2:

specify temporary tablesppace 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.

$
```

- Connect as HR in PDB21 and create the SQM as a scalar expression.

```

$ sqlplus hr@PDB21

Enter password: password

Connected to:

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>

```

- 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>

```

- 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>

```

Step 2 : Use SQL Macros as a table expression

- The first usage of a SQL macro as a table expression shows how to use the SQM to implement a polymorphic view.
 - 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>

```

- 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>
```

- 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>
```

- Use the SQM to display the result for departments 10 and 50.

```
SQL> SELECT * FROM budget() WHERE department_id IN (10,50);
```

DEPARTMENT_ID	BUDGET
50	156400
10	4400

```
SQL>
```

- The second usage of a SQL macro as a table expression shows how to use the SQM to display sum of the salaries per department for a particular job.
 - 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>

```

- 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>

```

- 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>

```

- Use the DBMS_OUTPUT package to display the rewritten SQL query. Re-create 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>
```

- 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>

```

- o 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
CONCAT_SELF	SCALAR	FUNCTION
SECURE_DML	NULL	PROCEDURE
ADD_JOB_HISTORY	NULL	PROCEDURE
BUDGET	TABLE	FUNCTION
BUDGET_PER_JOB	TABLE	FUNCTION
SECURE_EMPLOYEES		TRIGGER
UPDATE_JOB_HISTORY		TRIGGER

```

7 rows selected.

SQL> EXIT
$

```

Text

- Custom Range Bucketing in Result Set Interface
- Facet Navigation Support for JSON Search Indexes
- Improved Index Synchronization and Automatic Index Optimization
- In-Memory Full Text Columns
- JSON Support in Result Set Interface
- Named Entity Recognition Improvements
- New DIRECTORY_DATASTORE Data Store Type for Oracle Text
- New NETWORK_DATASTORE Data Store Type for Oracle Text
- New Oracle Text Index Type: Search Index

Custom Range Bucketing in Result Set Interface

You can now specify range buckets for faceted navigation. For example, you can group all items costing between \$10 and \$20.

Range buckets increase the usability of faceted navigation and avoid the need to manually create range-value sections in the indexed table.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

Facet Navigation Support for JSON Search Indexes

JSON search indexing now supports facet navigation using a JSON result set.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

Improved Index Synchronization and Automatic Index Optimization

Index synchronization performance is considerably improved, particularly when lots of sessions are doing DML operations simultaneously with the sync-on-commit option. Optimize index now optimizes all index entries (including SDATA values) rather than just the word-based index entries.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

In-Memory Full Text Columns

In previous releases, queries using `CONTAINS()` and `JSON_TEXTCONTAINS()` were only evaluated with a text index and JSON search index. Oracle Text technology now supports an `INMEMORY TEXT` clause during table creation, enabling fast in-memory searching rather than creating a separate on-disk text index.

An in-memory table scan can now evaluate columns directly when they are specified as `INMEMORY TEXT`. Queries using `CONTAINS()` or `JSON_TEXTCONTAINS()` can evaluate these operators in SQL predicates without requiring separate domain-specific indexes. When the In-Memory Column Store contains both scalar and non-scalar columns, On-line Transaction Processing (OLTP) applications that access both types of data can avoid accessing the row store, thereby improving performance.

Related Topics

- [Oracle® Database In-Memory Guide](#)

JSON Support in Result Set Interface

The JSON Result Set Interface (RSI) enables you to perform queries in JSON and return results as JSON.

The RSI enables you to fetch a set of results (a "hitlist") together with summary data such as the total number of hits and facet navigation information. This feature provides easier integration with modern programming languages which support JSON.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

Named Entity Recognition Improvements

Some restrictions have been removed from Named Entity Recognition (previously called Entity Extraction). Entity definitions can now be nested.

You can now incorporate existing or user-defined rules into other rules. For example, you can define rule "percentage change" as one of several words (for example, 'climbed', 'dropped', 'rose', 'increased') followed by the built-in rule for percentage.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

New DIRECTORY_DATASTORE Data Store Type for Oracle Text

You can now use a new data store type called `DIRECTORY_DATASTORE` instead of the deprecated `FILE_DATASTORE` type.

`DIRECTORY_DATASTORE` provides greater security because it enables file access to be based on directory objects.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

New NETWORK_DATASTORE Data Store Type for Oracle Text

You can now use a new data store type called `NETWORK_DATASTORE` instead of the deprecated `URL_DATASTORE` data type.

`NETWORK_DATASTORE` provides greater security because it enables URL access based on access control lists (ACLs), which support `HTTP` and `HTTPS` protocols.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

New Oracle Text Index Type: Search Index

The search indexes for text columns support sharding and all partition types. Index creation syntax is also simpler now.

The search index type maintains most of the features of the traditional `CONTEXT` index type, but has support for sharding and for multiple partitioning methods (the previous `CONTEXT` index type can only support range partitioning). It also has a simplified index creation syntax avoiding the semantic complexity necessary for `CONTEXT` indexes.

Related Topics

- [Oracle® Text Application Developer's Guide](#)

Big Data and Data Warehousing Solutions

- Analytic Views
- Analytical SQL and Statistical Functions
- Machine Learning for SQL
- Machine Learning for Python
- Query Optimization
- Spatial and Graph

Analytic Views

- [Analytic View Enhancements to SQL and PL/SQL](#)

Analytic View Enhancements to SQL and PL/SQL

SQL and PL/SQL enhancements for analytic views are the following:

- Star views can be created on fact and dimension tables
- Analytic views are created automatically for queries of the tables and views
- Level group caches are created and maintained autonomously
- Query-scoped base measures
- Support for remote sources
- Support for aggregate tables
- Star support for dimension caches

With these enhancements:

- Star view and star table queries can use the benefits of analytic views without any code changes
- Front-end tools can use analytic view calculations by inserting a new expression
- Level group caches can become more efficient over time based on usage patterns

Related Topics

- [Oracle® Database Data Warehousing Guide](#)

Analytical SQL and Statistical Functions

- [Bitwise Aggregate Functions](#)
- [Enhanced Analytic Functions](#)
- [New Analytical and Statistical Aggregate 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 SQL Language Reference](#)

Practice: Using Bitwise Aggregate Functions

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Test the bitwise AND function

- Connect to PDB21 as `SYSTEM` to query values with numbers and bitwise aggregate functions.

```

$ sqlplus system@PDB21
Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter password:

Connected to:

SQL>

```

- 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>

```

Step 2 : Test the bitwise OR function

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>

```

Step 3 : Test the bitwise XOR function

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
$
```

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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

The `setup_analytic_table.sh` shell script creates in both PDB21 and PDB19 the `REPORT` user, grants the `CREATE SESSION`, `CREATE TABLE` and `UNLIMITED TABLESPACE` privileges to `REPORT`, and creates and populates the `TRADES` table.

- Run the `setup_analytic_table.sh` script.

```
$ cd /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/setup_analytic_table.sh

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

Connected to:

SQL> DROP USER report CASCADE;
```

```
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 VA  
RCHAR2(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  
$
```

Step 2 : Experiment with the `GROUPS` clause of the window frame

- 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. `GROUPS` combines some of the features of both `ROWS` and `RANGE`.

```

$ sqlplus report@PDB21

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

Enter password:

Connected to:

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>

```

- Compute the total amount over the last five days on which account number 123 performed a "buy." To accomplish this, 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 PRE
              CEDING 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.

- 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 `GROUPS` keyword instead of `RANGE` or `ROWS`. The `GROUPS` keyword emphasizes the relationship to grouped queries. Using this kind of keyword, you can create 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 C
URRENT 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>
```



The aggregate function with the `DISTINCT` specification cannot be used with a window specification having a 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 C
URRENT 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 was the case in the previous step.

Step 3 : Experiment 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>
```

- Display the rows of table `m`

- Display the rows of table t.

```
SQL> SELECT * FROM t;

-----
V
-----
1
1
3
5
5
5
6

7 rows selected.

SQL>
```

- 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 EXCLU
DE CURRENT ROW) AS current_row,
           sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCLU
DE GROUP) AS the_group,
           sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCLU
DE TIES) AS ties,
           sum(v) OVER (o ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCLU
DE 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 EXCLU
DE CURRENT ROW) AS current_row,
           sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING EXCLU
DE GROUP) AS the_group,
           sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING EXCLU
DE TIES) AS ties,
           sum(v) OVER (o ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING EXCLU
DE 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>

- Use the EXCLUDE options for window frame exclusion with RANGE.

```

SQL> SELECT v,
           sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCL
           UDE CURRENT ROW) AS current_row,
           sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCL
           UDE GROUP) AS the_group,
           sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCL
           UDE TIES) AS ties,
           sum(v) OVER (o RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING EXCL
           UDE 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>

Step 4 : Experiment with the `GROUPS` and `EXCLUDE` clauses of the window frame

- 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
$

```

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 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`.

`ANY_VALUE`, a function to simplify and optimize the performance of `GROUP BY` statements, returns a random value in a group and is optimized to return the first value in the group. It ensures that there are no comparisons for any incoming row and eliminates the necessity to specify every column as part of the `GROUP BY` clause.

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 SQL Language Reference](#)

Practice: Detecting Data Tampering with the CHECKSUM Function

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Execute the `/home/oracle/labs/M104784GC10/setup_SH_tables.sh` shell script to create and load the `SH.SALES` and `SH.TIMES` tables.

```
$ cd /home/oracle/labs/M104784GC10
$ /home/oracle/labs/M104784GC10/setup_SH_tables.sh
...
Tablespace dropped.

Tablespace created.

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

Last Successful login time: Wed Mar 25 2020 03:18:51 +00:00

Connected to:

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_v3ctl
/home/oracle/labs/M104784GC10/sales_history/time_v3.dat
/home/oracle/labs/M104784GC10/time_v3.log

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 additional SALES using:
/home/oracle/labs/M104784GC10/sales_history/dmsal_v3ctl
/home/oracle/labs/M104784GC10/sales_history/dmsal_v3.dat
/home/oracle/labs/M104784GC10/dmsal_v3.log

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 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.

SQL>
```

Step 2 : Examine data before tampering

- 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.

```

$ sqlplus system@PDB21
Enter 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>

```

- 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>

```

- 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

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

Last Successful login time: Wed Mar 25 2020 03:20:17 +00:00

Connected to:

525 rows updated.

Commit complete.

$

```

Step 3 : Examine data after tampering

- 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 the earlier step, someone tampered the data.

- What happens if someone attempted to tamper with old sales? In the SH session, update some old sales and then roll back the transaction.

```

$ sqlplus sh@PDB21

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:

SQL> UPDATE sh.sales SET amount_sold = amount_sold*2 WHERE time_id='30-NOV-98
';

525 rows updated.

SQL> ROLLBACK;

Rollback complete.

SQL>

```

- 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 update was rolled back.

- Verify 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`. Note that `UNIQUE` is an Oracle-specific keyword and not 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>
```

- 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>
```

- 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 update was committed.

```
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 previously.

- How is NULL considered? Still in the initial terminal session, check that no one tampered with customer email addresses.
 - First, get the checksum value of the customer email addresses whose CUST_INCOME_LEVEL is equal to 300000 or higher.

```
SQL> SELECT cust_email FROM sh.customers WHERE cust_income_level = 'L: 300,000 and above';
```

```
CUST_EMAIL
```

```
-----
```

```
Gowen@company.example.com
```

```
Gowen@company.example.com
```

```
...
```

```
Krishnan@company.example.com
```

```
Prabu@company.example.com
```

```
1684 rows selected.
```

```
SQL> SELECT CHECKSUM(cust_email) FROM sh.customers;
```

```
CHECKSUM(CUST_EMAIL)
```

```
-----
```

```
107013
```

```
SQL> SELECT CHECKSUM(DISTINCT cust_email) FROM sh.customers;
```

```
CHECKSUM(DISTINCTCUST_EMAIL)
```

```
-----
```

```
227092
```

```
SQL>
```

- o In the SH session, set the customer email addresses to NULL for customers whose CUST_INCOME_LEVEL is equal to 300000 or higher.

```
SQL> UPDATE sh.customers SET cust_email = NULL
      WHERE cust_income_level = 'L: 300,000 and above';
```

```
1684 rows updated.
```

```
SQL> COMMIT;
```

```
Commit complete.
```

```
SQL>
```

- o In the initial terminal session, get the new checksum value of customer email addresses of customers whose CUST_INCOME_LEVEL is equal to 300000 or higher after the update.

```

SQL> SELECT DISTINCT cust_email FROM sh.customers WHERE cust_income_level = 'L: 300,000 and above';

CUST_EMAIL
-----

SQL> SELECT CHECKSUM(cust_email) FROM sh.customers;

CHECKSUM(CUST_EMAIL)
-----
                    577487

SQL> SELECT CHECKSUM(DISTINCT cust_email) FROM sh.customers;

CHECKSUM(DISTINCTCUST_EMAIL)
-----
                            141231

SQL> EXIT
$

```



Be aware that NULL values in the CHECKSUM column are ignored in external tables.

Practice: Measuring Asymmetry in Data with the SKEWNESS Functions

Overview

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`) is always deterministic.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Connect to PDB21 as HR and execute the `/home/oracle/labs/M104784GC10/Houses_Prices.sql` SQL script to create a table with skewed data.


```
$ cd /home/oracle/labs/M104784GC10
$ sqlplus system@PDB21

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

Enter password:
Last Successful login time: Mon Mar 16 2020 08:49:41 +00:00

Connected to:

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_ci
ty NUMBER, price_date DATE);

Table created.

SQL> INSERT INTO houses VALUES (1,100000,10000, sysdate);

1 row created.
...
SQL> INSERT INTO houses VALUES (1,900000,40000, sysdate+5);

1 row created.
...
SQL> COMMIT;

Commit complete.

SQL>
```

Step 2 : Examine skewed data

- Display the table rows. The `HOUSE` column values refer to types of houses that you want to look at and is used to categorize the data that you look at statistically and compare. 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>

```

- 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.

Step 3: Examine skewed data after data evolution

- Insert more rows into 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_cit
y) 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_c
ity) 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 the number of values in the data set increases, the difference between the computed values of `SKEWNESS_SAMP` and `SKEWNESS_POP` decreases.

- Determine the skewness of distinct values in the `PRICE_BIG_CITY` and `PRICE_SMALL_CITY` columns.

```
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 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.

- Insert more rows into 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
$

```

Practice: Measuring Tailedness of Data with the KURTOSIS Functions

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Connect to `PDB21` as `HR` and execute the `/home/oracle/labs/M104784GC10/Houses_Prices.sql` SQL script to create and populate a table.

```
$ cd /home/oracle/labs/M104784GC10
$ sqlplus system@PDB21

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

Enter password:
Last Successful login time: Mon Mar 16 2020 08:49:41 +00:00

Connected to:

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_ci
ty 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> COMMIT;

Commit complete.

SQL>
```

Step 2 : Examine the kurtosis of the distribution

- Display the table rows. The HOUSE column values refer to types of house that you want to look at and is used 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	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>

```

- 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.

Step 3 : Examine the kurtosis of the distribution after data evolution

- Insert more rows into 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_cit
y) 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_c
ity) 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.

- Determine the kurtosis of distinct values in `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 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.

- Insert more rows into 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.

Machine Learning for SQL

- Adam Optimization Solver for the Neural Network Algorithm
- Oracle Machine Learning MSET-SPRT Algorithm
- Oracle Machine Learning XGBoost Algorithm

Adam Optimization Solver for the Neural Network Algorithm

Adam is an optimization solver for the Neural Network algorithm that is computationally efficient, requires little memory, and is well suited for problems that are large in terms of data or parameters or both.

Adam is a popular extension to stochastic gradient descent. It uses mini-batch optimization and can make progress faster while seeing less data than the other Neural Network optimization solver, Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) with line search.

Related Topics

- [Oracle® Machine Learning for SQL Concepts](#)

Oracle Machine Learning MSET-SPRT Algorithm

The Multivariate State Estimation Technique-Sequential Probability Ratio Test (MSET-SPRT) algorithm is a nonlinear, nonparametric anomaly detection technique for monitoring critical processes.

The `DBMS_DATA_MINING.ALGO_MSET_SPRT` algorithm detects subtle anomalies while producing minimal false alarms. It calibrates expected behavior from available, historical data of the normal operational sequence of monitored signals. It incorporates the learned behavior of the system into a persistent MSET-SPRT model. You can apply the model to new records to detect anomalous behavior.

Related Topics

- [Oracle® Machine Learning for SQL Concepts](#)

Oracle Machine Learning XGBoost Algorithm

XGBoost is a highly-efficient, scalable gradient tree boosting machine learning algorithm for regression and classification.

The `DBMS_DATA_MINING.ALGO_XGBOOST` algorithm prepares training data, builds and persists a model, and applies the model for prediction. You can use it as a stand-alone predictor or incorporate it into real-world production pipelines for a wide range of problems such as ad click-through rate prediction, hazard risk prediction, web text classification, and so on.

Related Topics

- [Oracle® Machine Learning for SQL Concepts](#)

Machine Learning for Python

- [Oracle Machine Learning for Python \(OML4Py\)](#)
- [Oracle Machine Learning for Python Configuration in DBCA](#)

Oracle Machine Learning for Python (OML4Py)

Oracle Machine Learning for Python (OML4Py) enables the open source Python programming language and environment to operate on database data at scale. Python users can run Python commands and scripts for statistical analysis and machine learning on data stored in Oracle Database.

With OML4Py, you can do the following:

- Use a wide range of in-database machine learning algorithms
- Minimize data movement
- Leverage Oracle Database as a high performance compute engine for data exploration and preparation
- Use AutoML for automatic algorithm selection, feature selection, and model tuning
- Execute user-defined Python functions in non-parallel, data-parallel, and task-parallel fashion

[Details: AutoML in OML4Py](#)

This page provides more detailed information about using OML4Py AutoML.

Related Topics

- [Oracle® Machine Learning for Python](#)

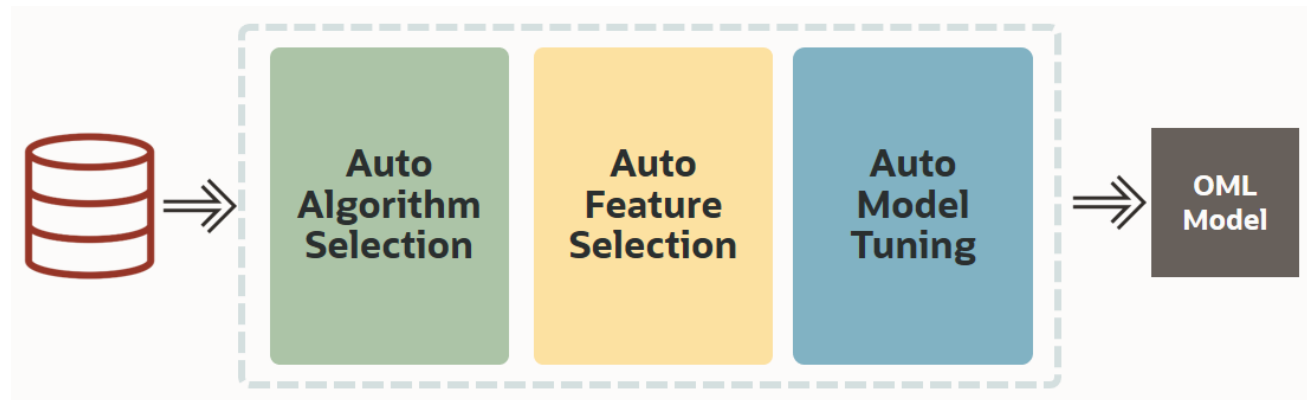
Details: AutoML in OML4Py

This page provides more detailed information about using OML4Py AutoML.

OML4Py AutoML aims to eliminate repetitive tasks such as model building and evaluation through automation, with a goal of increasing user productivity for both expert data scientists and non-experts.

AutoML applies machine learning to the machine learning process itself, using what we call meta-learning to automatically zero in on the right algorithm, features, and algorithm settings necessary to build high-quality models without resorting to exhaustive search. This automation helps to make machine learning more accessible to a broader audience, while reducing the compute time and cost required to get to the desired result.

AutoML increases data scientist productivity, while reducing overall compute time, and enables non-expert Python users to leverage machine learning – in part because they don't need to be experts in the modeling process, the range of algorithms with their data preparation requirements, or understand each algorithm's hyperparameters.



OML4Py's AutoML consists of three steps, as shown in the diagram:

1. **Auto Algorithm Selection:** With automated algorithm selection, the goal is to identify the in-database algorithms likely to achieve the highest model quality. Using meta-learning, AutoML leverages machine learning itself to help find the best algorithm faster than with exhaustive search.
2. **Auto Feature Selection:** With automated feature selection, the goal is to de-noise data by eliminating features that don't add value to the model. By identifying the most predictive features and eliminating noise, model accuracy can often be significantly improved, with the side benefit of faster model building and scoring.
3. **Auto Model Tuning:** Automated model tuning tunes algorithm hyperparameters and can significantly improve accuracy while avoiding manual or exhaustive search techniques, which can be costly both in terms of time and compute resources

Oracle Machine Learning for Python Configuration in DBCA

The Database Configuration Assistant (DBCA) supports configuring Oracle Machine Learning for Python (OML4Py).

Configuration through DBCA is supported for container databases and pluggable databases.

Related Topics

- [Oracle® Multitenant Administrator's Guide](#)

Query Optimization

- [In-Memory Vectorized Joins](#)

In-Memory Vectorized Joins

The In-Memory vectorized joins feature is based on the deep vectorization framework. Using SIMD vector processing, the framework optimizes aspects of hash joins such as hashing, building, probing, and gathering. This optimization can improve the performance of join processing by 100% or more. The In-Memory vectorized joins feature is transparent to the user, requiring no plan changes.

In-Memory deep vectorization is a SIMD-based query processing framework that supports vectorization for higher-level query operators in the query plan. The framework includes optimizations such as SIMD, hardware acceleration and pipelined execution.

Joins can account for a large percentage of SQL execution time for data warehouse workloads. Improving performance of hash joins by 100% or more can significantly improve performance.

[Details: In-Memory Deep Vectorization](#)

This page provide more detailed information about In-Memory deep vectorization.

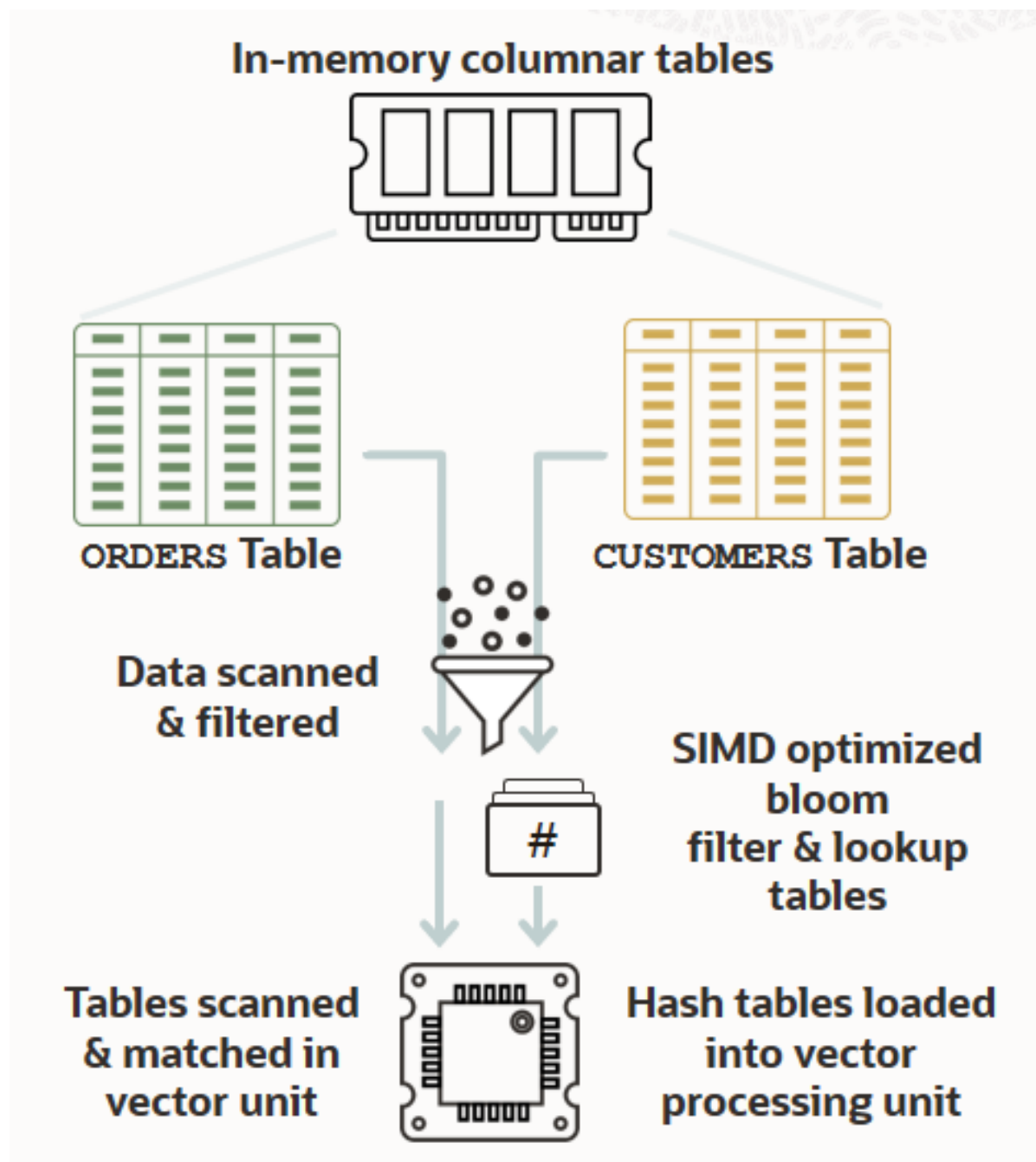
Related Topics

- [Oracle® Database In-Memory Guide](#)

Details: In-Memory Deep Vectorization

This page provides more detailed information about In-Memory deep vectorization.

In-Memory deep vectorization is a SIMD-based framework that supports vectorization for higher-level query operators in the query plan. The framework includes optimizations such as SIMD, hardware acceleration, and pipelined execution. The In-Memory vectorized joins feature is transparent to the user, requiring no plan changes.



The In-Memory vectorized joins feature is key to the deep vectorization framework. Using SIMD vector processing, the framework optimizes aspects of hash joins such as hashing, building, probing, and gathering. This optimization can improve the performance of join processing by 100% or more.

The In-Memory deep vectorization framework deconstructs high-level, complex SQL operators such as hash joins into smaller kernel-sized units. The deconstructed kernels are suitable for SIMD vectorization techniques. The database executes the kernels in a pipelined fashion to accelerate the overall operation.

Spatial and Graph

- [Property Graph: Graph Server and Client Kit](#)
- [Property Graph: Native Python Client](#)
- [Property Graph: New Features in PGQL](#)
- [Property Graph: Optimized Graph Representation for Faster Performance](#)
- [Property Graph: User-defined Graph Algorithms](#)
- [Property Graph Visualization](#)
- [RDF Graph: Native Unicode Storage and Processing](#)
- [Self-service Low-code Spatial Studio](#)
- [Spatial Network Data Model Contraction Hierarchy](#)
- [Spatial Support for Database In-Memory](#)

Property Graph: Graph Server and Client Kit

You no longer have to copy the Property Graph in-memory analytics server (PGX) and client tools and libraries from `$ORACLE_HOME`.

With the simplified packaging and the availability of the Oracle Graph Server and Client kit on OCI Marketplace, application developers can quickly and securely install and deploy the components required to work with Property Graphs. This makes it easier to start developing applications.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

Property Graph: Native Python Client

Property Graph has a native Python API that allows you to create a graph, run graph queries (using PGQL), and analyze the graph using Python and Jupyter notebooks.

This makes it easier to get started with graph and simplifies integration in data science applications where Python and Jupyter are especially popular.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

Property Graph: New Features in PGQL

You can now do graph DDL and graph DML operations with the graph query language PGQL. Additionally, you can use PGQL for CHEAPEST path queries using cost functions.

This simplifies development by eliminating the need for configuration files and Java APIs when doing several common graph operations.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

Property Graph: Optimized Graph Representation for Faster Performance

The In-memory analytics server (PGX, or Property Graph AnalytiX) has an optimized representation of a property graph that uses less memory. Larger graphs can be analyzed in the same amount of memory.

Graph sizes are continuously growing larger. With this optimization you can analyze larger graphs using less memory than previously required. This not only enables analysis of more data, but also reduces system costs. The optimized graph representation gives you faster performance and is transparent; existing applications will run faster with no change.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

Property Graph: User-defined Graph Algorithms

You can create or extend graph algorithms using Java syntax, in addition to the dozens of pre-built graph analytics that come with the product. These user-defined algorithms will execute as fast as native algorithms in the product because they are compiled with the same optimizations.

For unique and specialized use cases, customizing graph algorithms lets you add analysis that analysts and data scientists design specifically for your applications.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

Property Graph Visualization

A rich set of visualization features lets you interactively explore the graph, customize layouts, and highlight interesting relationships in your data.

Seeing graph data and relationships visually lets analysts, data scientists, and developers quickly understand and explore clusters, outliers, anomalies, patterns, communities, and critical connections in their data. This makes you more productive and helps you share and communicate results more clearly.

Related Topics

- [Oracle Spatial and Graph Property Graph Developer's Guide](#)

RDF Graph: Native Unicode Storage and Processing

When creating the RDF (Resource Description Framework) network, you can now store RDF data in a native unicode format for virtually all use cases.

This reduces the storage required and enhances query performance.

Related Topics

- [Oracle® Database Graph Developer's Guide for RDF Graph](#)

Self-service Low-code Spatial Studio

Oracle Spatial Studio is a self-service web application that makes it easy for you to create interactive maps and perform spatial analysis on your business data. You can also use Spatial Studio to publish spatial analyses as REST services and generate SQL statements for spatial analysis using low-code and UI components.

With Spatial Studio, you no longer need to write Javascript or SQL or use third-party tools to take advantage of the spatial capabilities in Oracle Database.

Related Topics

- [Oracle® Spatial Developer's Guide](#)

Spatial Network Data Model Contraction Hierarchy

Contraction hierarchy, a precomputed in-memory approach, speeds up path computations in the Oracle Spatial and Graph network data model.

By using the contraction hierarchy Java API, you can evaluate shortest path computations, drive time polygon analysis, and traveling salesperson analysis functions more efficiently. These functions will perform 10 to 100 times faster than with previous releases. You can run more network analysis functions using fewer CPUs, and support more concurrent requests using the same hardware.

Related Topics

- [Oracle® Topology and Network Data Model Developer's Guide](#)

Spatial Support for Database In-Memory

A new In-Memory Spatial index is created when the `INMEMORY SPATIAL` clause is specified for a spatial geometry column during a `CREATE` or `ALTER TABLE` statement.

This means that you no longer have to create and maintain conventional disk-based spatial indexes when spatial tables are stored using Database In-Memory. The in-memory spatial index provides much faster query performance for `SDO_FILTER()` operations without having to maintain a separate, on-disk spatial index.

Related Topics

- [Oracle® Spatial Developer's Guide](#)

Database Upgrade and Utilities

- [Database Utilities](#)
- [Upgrades and Migration](#)

Database Utilities

- Oracle Data Pump Checksums Support Cloud Migrations
- Oracle Data Pump Exports from Oracle Autonomous Database
- Oracle Data Pump Includes and Excludes in the Same Operation
- Oracle Data Pump Parallelizes Transportable Tablespace Metadata Operations
- Oracle Data Pump Provides Optional Index Compression
- Oracle Data Pump Resumes Transportable Tablespace Jobs
- Oracle Data Pump Supports Export to and Import From Cloud Object Stores
- Oracle Data Pump Supports Native JSON Datatypes
- Oracle SQL *Loader Support for Object Store Credentials
- Oracle SQL *Loader Supports Native JSON Data Type

Oracle Data Pump Checksums Support Cloud Migrations

To check Oracle Data Pump dump files for validity, you can now use checksums that are added to the dump file.

Oracle Data Pump is used for migrating application data from on-premises Oracle Database instances into the Oracle Cloud, and also for copying dump files to on-premises.

In this release, a checksum is now added to the dump file. 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 dump files 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

Overview

This practice shows how to use a 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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Execute the `/home/oracle/labs/M104786GC10/DP.sh` shell script. The shell script creates the `HR.EMPLOYEES` table to use in an export on `PDB21`.

```
$ cd /home/oracle/labs/M104786GC10
$ /home/oracle/labs/M104786GC10/DP.sh
...
specify password for HR as parameter 1:

specify default tablespace 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.
...
***** Creating EMPLOYEES table ....

Table created.

Index created.

Table altered.

Table altered.

Sequence created.
...
Commit complete.

Session altered.
...
***** Populating EMPLOYEES table ....

1 row created.
...

Commit complete.

Index created.
...
Commit complete.

Procedure created.

Trigger created.

Trigger altered.

Procedure created.

Trigger created.

Commit complete.
...
Directory created.

Grant succeeded.

$
```

Step 2 : Export the table using the checksum

- Export the `HR.EMPLOYEES` table 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 21c 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. Use the `CHECKSUM` parameter during the export operation.

```
$ expdp system@PDB21 TABLES=hr.employees DUMPFILE=dp_dir:emp.dmp CHECKSUM=yes
REUSE_DUMPFILES=yes
```

```
Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:
```

```
Starting "SYSTEM"."SYS_EXPORT_TABLE_01": system/*****@PDB21 TABLES=hr.emp
loyees 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 row
s
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, a 256-bit hash algorithm.

- If you want to use the SHA384 384-bit hash algorithm, the SHA512 512-bit hash algorithm, or the CRC32

32-bit checksum, use the `CHECKSUM_ALGORITHM` parameter and not the `CHECKSUM` parameter. The `CHECKSUM` parameter uses the SHA256 256-bit hash algorithm.

```
$ expdp system@PDB21 TABLES=hr.employees DUMPFILE=dp_dir:emp384.dmp CHECKSUM_
ALGORITHM=SHA384 CHECKSUM=no REUSE_DUMPFILES=yes
```

```
Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:
```

```
ORA-39002: invalid operation
```

```
ORA-39050: parameter CHECKSUM=NO is incompatible with parameter CHECKSUM_ALGO
RITHM
```

```
$
```

```
$ expdp system@PDB21 TABLES=hr.employees DUMPFILE=dp_dir:emp512.dmp CHECKSUM_
ALGORITHM=SHA512 REUSE_DUMPFILES=yes
```

```
Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:
```

```
Starting "SYSTEM"."SYS_EXPORT_TABLE_01": system/*****@PDB21 TABLES=hr.emp
loyees 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 row
s
```

```
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
```

```
$
```

Step 3 : Import the table

- Drop the table before importing it.

```
$ sqlplus hr@PDB21

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

Enter password:

Connected to:

SQL> DROP TABLE employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
$
```

- Before importing the table, determine whether the dump files are corrupted or not.
 - Corrupt one of the dump files by executing the `/home/oracle/labs/M104786GC10/corrupt.sh` shell script.

```
$ /home/oracle/labs/M104786GC10/corrupt.sh
$
```

- Determine which of the two dump files is corrupted.

```
$ impdp system@PDB21 FULL=yes DUMPFILE=dp_dir:emp512.dmp VERIFY_ONLY=YES

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

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@PDB21 FULL=yes DUMPFILE=dp_dir:emp.dmp VERIFY_ONLY=YES

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39411: header checksum error in dump file "/home/oracle/labs/M104786
GC10/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 head
er
//          was tampered with or otherwise corrupted due to transmission
or
//          media failure.
// *Action: Contact Oracle Support Services.
$

```

- Import the table.
 - 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@PDB21 FULL=yes DUMPFILE=dp_dir:emp.dmp

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39411: header checksum error in dump file "/home/oracle/labs/M104786
GC10/emp.dmp"
$

```

- 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 include the `VERIFY_CHECKSUM` parameter. Ignore the error messages related to indexes creation. The point of this practice is that the table can be reimported.

```
$ impdp system@PDB21 FULL=yes DUMPFILE=dp_dir:emp512.dmp VERIFY_CHECKSUM=
YES
```

```
Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
```

```
Password:
```

```
Verifying dump file checksums
```

```
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
```

```
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/*****@PDB21 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 10
```

```
7 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 0
```

```
7:20:29 2020 elapsed 0 00:00:20
```

```
$
```

- o Import using the non-corrupted dumpfile, avoiding the verification. Drop the table first.

```

$ sqlplus hr@pdb21

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

Enter password:

Connected to:

SQL> DROP TABLE employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT

$ impdp hr@PDB21 FULL=yes DUMPFILE=dp_dir:emp512.dmp VERIFY_CHECKSUM=NO

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
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/*****@PDB21 FULL=yes DUMP
FILE=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
$

```

Oracle Data Pump Exports from Oracle Autonomous Database

Starting with this release, Oracle Data Pump can perform exports from Oracle Autonomous Database into dump files in a cloud object store.

Oracle Data Pump supports dump file export from cloud services to the object store. You can now migrate data from services you manage in Oracle Autonomous Database.

Related Topics

- [Oracle® Database Database Utilities](#)

Oracle Data Pump Includes and Excludes in the Same Operation

Starting with this release, 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](#)

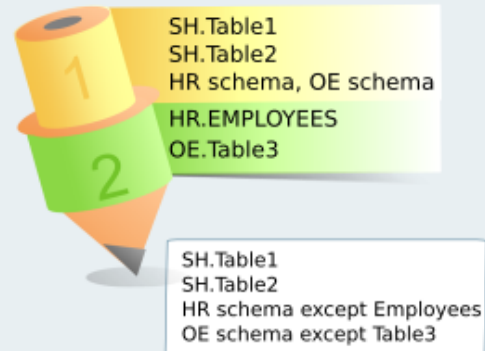
Details: Oracle Data Pump Includes and Excludes in the Same Operation

This page provides more detailed information about excluding and including objects with Oracle Data Pump export or import in a single command.

19c The `EXCLUDE` and `INCLUDE` parameters are mutually exclusive.

21c Including and excluding objects is possible in the same export / import operation:

1. Data Pump processes the `INCLUDE` parameter first and includes all objects identified.
2. Then Data Pump processes the `EXCLUDE` parameters. Any objects specified by the `EXCLUDE` parameter being in the list of `INCLUDE` objects are removed.



```
$ expdp ... INCLUDE = TABLE:"IN ('SH.TABLE1', 'SH.TABLE2')"  
INCLUDE = SCHEMA:"IN ('HR', 'OE')"  
EXCLUDE = TABLE:"IN ('HR.EMPLOYEES', 'OE.TABLE3')"
```

Starting with Oracle Database 21c, 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

Overview

This practice shows how to export or import objects by including and excluding objects during the same operation.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Use the `/home/oracle/labs/M104780GC10/create_PDB21_2.sh` shell script to create the `PDB21_2` PDB and the `HR` user in `PDB21_2`.


```
$ cd /home/oracle/labs/M104780GC10
$ /home/oracle/labs/M104780GC10/create_PDB21_2.sh
...
SQL> CREATE PLUGGABLE DATABASE pdb21_2 FROM pdb21 KEYSTORE IDENTIFIED BY pass
word;

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb21_2 OPEN;

Pluggable database altered.

SQL> exit

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

Connected to:

SQL> DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-12919: Can not drop the default permanent tablespace

SQL> CREATE TABLESPACE users;
CREATE TABLESPACE users
*
ERROR at line 1:
ORA-01543: tablespace 'USERS' already exists

SQL> DROP USER hr CASCADE;

User dropped.

SQL> CREATE USER hr IDENTIFIED BY password;

User created.

...
SQL> DROP TABLE hr.departments CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
^
```

- Before exporting the two HR tables, excluding their statistics, verify that statistics have been collected on the tables. Create a directory for the export dumpfile.
 - Verify that statistics have been collected for the two HR tables.

```
$ sqlplus system@PDB21

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

Enter password:
Last Successful login time: Tue Mar 17 2020 02:23:18 +00:00

Connected to:

SQL> SELECT num_rows FROM dba_tables WHERE table_name IN ('JOBS', 'DEPART
MENTS');

  NUM_ROWS
-----
         27
         19

SQL>
```

- Create a directory for the export dumpfile.

```
SQL> CREATE OR REPLACE DIRECTORY dp_dir AS '/home/oracle/labs';

Directory created.

SQL> GRANT read, write ON DIRECTORY dp_dir TO hr;

Grant succeeded.

SQL> EXIT
$
```

Step 2 : Export tables excluding their statistics

- From PDB21 export the two HR tables, excluding their statistics.

```

$ expdp hr@PDB21 DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:"IN \('JOBS
\','DEPARTMENTS'\)" EXCLUDE=STATISTICS REUSE_DUMPFILES=YES

Password:

Starting "HR"."SYS_EXPORT_SCHEMA_01": hr/*****@PDB21 DUMPFILE=hr.dmp DIRE
CTORY=dp_dir INCLUDE=TABLE:"IN ('JOBS','DEPARTMENTS')" EXCLUDE=STATISTICS REU
SE_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 row
s
. . exported "HR"."DEPARTMENTS"                        7.125 KB          27 row
s
ORA-39173: Encrypted data has been stored unencrypted in dump file set.
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
$

```

Step 3 : Import tables

- Import the dumpfile into another PDB, PDB21_2 in CDB21.

```

$ impdp system@PDB21_2 DUMPFILE=hr.dmp DIRECTORY=DP_DIR FULL=Y

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/*****@PDB21_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 row
s
. . imported "HR"."DEPARTMENTS"        7.125 KB      27 row
s
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
...
$

```

The import may complete with errors due to missing constraints for HR.DEPARTMENTS that reference other HR tables.

- Re-execute the export operation excluding statistics and constraints.

```

$ expdp hr@PDB21 DUMPFILE=hr.dmp DIRECTORY=dp_dir INCLUDE=TABLE:"IN \('JOBS
\','DEPARTMENTS'\)" EXCLUDE=STATISTICS,CONSTRAINT REUSE_DUMPFILES=YES

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

Starting "HR"."SYS_EXPORT_SCHEMA_01": hr/*****@PDB21 DUMPFILE=hr.dmp DIRE
CTORY=dp_dir INCLUDE=TABLE:"IN ('JOBS','DEPARTMENTS')" EXCLUDE=STATISTICS,CON
STRAINT 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 row
s
. . exported "HR"."DEPARTMENTS"                        7.125 KB          27 row
s
ORA-39173: Encrypted data has been stored unencrypted in dump file set.
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
$

```

- After dropping the HR.JOBS and HR.DEPARTMENTS tables, re-import the dumpfile into PDB21_2 in CDB2 1.

```

$ sqlplus system@PDB21_2

Enter password:

SQL> DROP TABLE hr.jobs CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE hr.departments CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT

$ impdp system@PDB21_2 DUMPFILE=hr.dmp DIRECTORY=DP_DIR FULL=Y

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/*****@PDB21_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 row
s
. . imported "HR"."DEPARTMENTS"        7.125 KB      27 row
s
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
...
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed at Tue Mar 17 04:03:
37 2020 elapsed 0 00:00:05
$

```



Observe that the import does not issue errors related to constraints. Constraints that should have been added to the HR.DEPARTMENTS table were excluded.

- Verify that statistics for the HR.JOBS and HR.DEPARTMENTS tables were also excluded.

```
$ sqlplus system@PDB21_2

Enter password:

SQL> SELECT num_rows FROM dba_tables WHERE table_name IN ('JOBS','DEPARTMENTS
');

  NUM_ROWS
-----

no rows selected

SQL> EXIT
$
```

Oracle Data Pump Parallelizes Transportable Tablespace Metadata Operations

Starting with this release, 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.

19c Restart a failed export / import operation, not a transportable tablespace operation

21c Resume a failed transportable tablespace export at or near the point of failure

19c Parallelize an export / import operation, not a transportable tablespace operation

21c Parallelize transportable tablespace metadata operations

```
$ expdp ... PARALLEL=2 TRANSPORT_TABLESPACES=tbs_1 TRANSPORT_FULL_CHECK=YES
```

```
$ impdp ... PARALLEL=2 TRANSPORT_DATAFILES='/user01/data/tbs1.dbf'
```

Starting with Oracle Database 21c, 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

Overview

The practice shows how to parallelize export and import operations for Transportable Tablespace (TTS) metadata.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Prepare the tablespace to be exported

- In PDB21, set the `USERS` tablespace to read only. If the tablespace does not exist, create it.

```

$ sqlplus sysPDB21 AS SYSDBA

Enter password:

Connected to:
SQL> CREATE TABLESPACE users;

Tablespace created.

SQL> ALTER TABLESPACE users READ ONLY;

Tablespace altered.

SQL> EXIT
$

```

Step 2 : Perform the TTS in parallel

- Perform the TTS in parallel against PDB21.

```

$ expdp \"sys@PDB21 AS SYSDBA\" dumpfile=PDB21.dmp TRANSPORT_TABLESPACES=users
s TRANSPORT_FULL_CHECK=YES LOGFILE=tts.log REUSE_DUMPFILES=YES PARALLEL=2

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.
Password:

Starting \"SYS\".\"SYS_EXPORT_TRANSPORTABLE_02\": \"sys/*****@PDB21 AS SYSDBA\"
dumpfile=PDB21.dmp TRANSPORT_TABLESPACES=users TRANSPORT_FULL_CHECK=YES LOGFI
LE=tts.log REUSE_DUMPFILES=YES PARALLEL=2
ORA-39396: Warning: exporting encrypted data using transportable option witho
ut password

ORA-39396: Warning: exporting encrypted data using transportable option witho
ut password

Processing object type TRANSPORTABLE_EXPORT/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type TRANSPORTABLE_EXPORT/INDEX/STATISTICS/BITMAP_INDEX/IND
EX_STATISTICS
Processing object type TRANSPORTABLE_EXPORT/STATISTICS/TABLE_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/TABLE
Processing object type TRANSPORTABLE_EXPORT/INDEX/INDEX
Processing object type TRANSPORTABLE_EXPORT/COMMENT
Processing object type TRANSPORTABLE_EXPORT/CONSTRAINT/CONSTRAINT
Processing object type TRANSPORTABLE_EXPORT/CONSTRAINT/DEF_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/INDEX
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/SECONDARY_TABLE
E/TABLE
Processing object type TRANSPORTABLE_EXPORT/INDEX/DOMAIN_INDEX/SECONDARY_TABLE
E/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/B33495ED418D1C83E0538705F40AD599/PDB21.dmp
*****
*
Datafiles required for transportable tablespace USERS:
  /u02/app/oracle/oradata/CDB21_fra1xn/CDB21_FRA1XN/B33495ED418D1C83E0538705F
40AD599/datafile/o1_mf_users_hx3vo2r5_.dbf
Job "SYS"."SYS_EXPORT_TRANSPORTABLE_02" completed with 2 error(s) at Tue Dec
15 08:15:16 2020 elapsed 0 00:02:32
$

```

Observe the two ORA-39396 messages during the export operation. This means that the two parallel operations encountered the same warning message.

Step 3 : Set the tablespace back to read write

- Use the `ALTER TABLESPACE` command to set the tablespace back to read write.

```

$ sqlplus sys@PDB21 AS SYSDBA

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

Enter password:

Connected to:

SQL> ALTER TABLESPACE users READ WRITE;

Tablespace altered.

SQL> EXIT
$

```


Oracle Data Pump Provides Optional Index Compression

In this release, 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 21c 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.

19c Add, change, or eliminate the table compression clause during import using a `TRANSFORM` parameter of `TABLE_COMPRESSION_CLAUSE`

```
$ impdp ... TRANSFORM=TABLE_COMPRESSION_CLAUSE:
\COLUMN STORE COMPRESS FOR QUERY HIGH\
```

```
$ impdp ... TRANSFORM=TABLE_COMPRESSION_CLAUSE: BASIC
```

21c Add, change, or eliminate the index compression clause during import using a `TRANSFORM` parameter of `INDEX_COMPRESSION_CLAUSE`

```
$ impdp ... TRANSFORM=INDEX_COMPRESSION_CLAUSE:\COMPRESS ADVANCED LOW\
```

```
$ impdp ... TRANSFORM=INDEX_COMPRESSION_CLAUSE:NONE
```

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 parameter changes the type of compression for all indexes in the job.

Practice: Using Index Compression on Import

Overview

The practice shows how to use index compression on import operations.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- Create the HR schema. Change the *password* string in the command to your password.

```
$ sqlplus "sys@PDB21 AS SYSDBA" @/home/oracle/labs/M104780GC10/hr_main.sql pa
ssword users temp /tmp

...
Commit complete.

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

- Verify that the HR.EMPLOYEES table is not using compression and owns indexes that are not using compression.

```
$ sqlplus SYSTEM@PDB21
Enter password:

Connected to:

SQL> SELECT compression, compress_for FROM DBA_TABLES WHERE table_name='EMPLO
YEES';

COMPRESSION COMPRESS_FOR
-----
DISABLED

SQL> COL INDEX_NAME FORMAT A30
SQL> SELECT index_name, compression FROM dba_indexes WHERE table_name='EMPLOY
EES';

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>
```

- Create a directory for Oracle Data Pump dumpfiles.

```
SQL> CREATE OR REPLACE DIRECTORY dp_dir AS '/home/oracle/labs';

Directory created.

SQL> GRANT read, write ON DIRECTORY dp_dir TO hr;

Grant succeeded.

SQL> EXIT
$
```

Step 2 : Export the table

- Export the HR.EMPLOYEES table. Ignore any Database Vault warning.

```
$ expdp hr@PDB21 DUMPFILE=PDB21.dmp DIRECTORY=dp_dir TABLES=EMPLOYEES REUSE_D
UMPFILES=YES

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

Starting "HR"."SYS_EXPORT_TABLE_01": hr/*****@PDB21 DUMPFILE=PDB21.dmp DI
RECTORY=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 row
s
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:
  /home/oracle/labs/PDB21.dmp
Job "HR"."SYS_EXPORT_TABLE_01" successfully completed at Tue Dec 15 08:29:22
2020 elapsed 0 00:00:26
$
```


Step 3 : Import the table using the compression parameters

- Drop the table in PDB21.

```

$ sqlplus SYSTEM@PDB21

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

Last Successful login time: Wed Apr 08 2020 16:24:56 +00:00

Connected to:

SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;

Table dropped.

SQL> EXIT
$

```

- Import the table using the index compression and the table compression parameters.

```

$ impdp hr@PDB21 FULL=Y DUMPFILE=PDB21.dmp DIRECTORY=dp_dir TRANSFORM=TABLE_C
OMPRESSION_CLAUSE:\"COMPRESS BASIC\" TRANSFORM=INDEX_COMPRESSION_CLAUSE:\"COM
PRESS ADVANCED LOW\" EXCLUDE=CONSTRAINT

Copyright (c) 1982, 2020, Oracle and/or its affiliates. All rights reserved.
Password:

Master table "HR"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "HR"."SYS_IMPORT_FULL_01": hr/*****@PDB21 FULL=Y DUMPFILE=PDB21.
dmp DIRECTORY=dp_dir TRANSFORM=TABLE_COMPRESSION_CLAUSE:"COMPRESS BASIC" TRAN
SFORM=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 row
s
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
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" successfully completed at Tue Dec 15 08:30:27 2
020 elapsed 0 00:00:32
$

```

- Verify that the imported table is using compression and that its indexes also use compression.

```

$ sqlplus SYSTEM@PDB21

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

Last Successful login time: Wed Apr 08 2020 16:38:57 +00:00

Connected to:

SQL> SELECT compression, compress_for FROM DBA_TABLES WHERE table_name='EMPLOY
YEES';

COMPRESS COMPRESS_FOR
-----
ENABLED BASIC

SQL> COL INDEX_NAME FORMAT A30
SQL> SELECT index_name, compression FROM dba_indexes WHERE table_name='EMPLOY
EES';

INDEX_NAME                                COMPRESSION
-----
EMP_DEPARTMENT_IX                         ADVANCED LOW
EMP_JOB_IX                                 ADVANCED LOW
EMP_MANAGER_IX                             ADVANCED LOW
EMP_EMP_ID_PK                              DISABLED
EMP_NAME_IX                                 ADVANCED LOW

SQL> EXIT
$

```

- Why is the primary key index not compressed?

```
$ sqlplus SYSTEM@PDB21
```

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

```
Last Successful login time: Wed Apr 08 2020 16:24:56 +00:00
```

```
Connected to:
```

```
SQL> ALTER INDEX hr.emp_emp_id_pk REBUILD COMPRESS ADVANCED LOW;
```

```
ALTER INDEX hr.emp_emp_id_pk REBUILD COMPRESS ADVANCED LOW
```

```
*
```

```
ERROR at line 1:
```

```
ORA-25193: cannot use COMPRESS option for a single column key
```

```
SQL> EXIT
```

```
$
```

Oracle Data Pump Resumes Transportable Tablespace Jobs

Starting with this release, 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 Supports Export to and Import From Cloud Object Stores

Starting with Oracle Database 21c, Oracle Data Pump `expdp` and `impdp` support use of object store URIs for the `DUMPFILE` parameter.

To use this feature for exports or imports from an object store, the `CREDENTIAL` parameter must be set. This feature eases migration to and from Oracle Cloud, because it relieves you of the extra step of transferring a dumpfile to or from the object store. Note that export and import performance is slower when accessing the object store, compared to local disk access, but the process is simpler. In addition, the process should be faster than running two separate export operations from Oracle Cloud, and transferring the dumpfile from the object store to an on premises location, or transferring the dumpfile from on premises to the object store, and then importing into Oracle Cloud.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Oracle Data Pump Supports Native JSON Datatypes

In this release, Oracle Data Pump enables export and import of Oracle Database native JSON objects.

Oracle Data Pump supports export and import of the new native JSON datatype in Oracle Database. This includes conventional and Transportable Tablespace (TTS) export and import of tables containing the JSON Datatype in full, tablespaces and table modes.

Oracle SQL*Loader Support for Object Store Credentials

Starting with this release, Oracle SQL*Loader accesses data in an object store by presenting user-defined credentials.

Oracle SQL*Loader now enables you to specify credentials that you define, so that you can access and read data from files in an object store into Oracle Database.

Related Topics

- [Oracle® Database Database Utilities](#)

Oracle SQL*Loader Supports Native JSON Data Type

Oracle SQL*Loader enables JSON file data loading into Oracle Database native JSON objects.

Starting with this release, SQL*Loader conventional and direct path supports loading JSON data into the new native JSON data type in Oracle Database.

Related Topics

- [Oracle® Database Database Utilities](#)

Upgrades and Migration

- [AutoUpgrade Automates Data Guard Operations During Database Upgrade](#)
- [AutoUpgrade Automates Steps Required for Oracle RAC Database Upgrade](#)
- [AutoUpgrade Automates Upgrade and Conversion of Non-CDB to PDB](#)
- [AutoUpgrade Automates Upgrade of a PDB via Unplug-Plug-Upgrade](#)
- [Oracle Database Automates Database Upgrades with AutoUpgrade](#)

AutoUpgrade Automates Data Guard Operations During Database Upgrade

Starting with Oracle Database 21c, AutoUpgrade automates the steps necessary to upgrade a database in an Oracle Data Guard configuration.

Steps that AutoUpgrade automates includes disabling and re-enabling the Data Guard Broker (if present), and deferring log file transport while the upgrade takes place. Automation of these Oracle Data Guard operations in the context of a database upgrade eliminates the need for manual or scripted steps that you otherwise would have to perform yourself. This in turn makes it easier to automate upgrades of databases in Oracle Data Guard configurations. This feature particularly helps to automate upgrades in high-end environments, where there can be many hundreds of such databases that you upgrade as part of your upgrade project.

Related Topics

- [Oracle® Database Database Upgrade Guide](#)

AutoUpgrade Automates Steps Required for Oracle RAC Database Upgrade

In Oracle Database 21c, AutoUpgrade automates all the steps needed to upgrade an Oracle Real Application Clusters (Oracle RAC) database.

AutoUpgrade automates steps that previously had to be performed manually or scripted. Steps that AutoUpgrade automates include the `SRVCTL` commands to stop and restart instances on multiple nodes, handling of `CLUSTER_DATABASE`, updating initialization parameter files, and restarting services, where appropriate.

This feature reduces the possibility of manual errors and decreases the work needed for DBAs to perform routine maintenance and upgrade activities.

Related topics

- [Oracle® Database Database Upgrade Guide](#)

AutoUpgrade Automates Upgrade and Conversion of Non-CDB to PDB

To simplify upgrades to Oracle Database 21c, AutoUpgrade automates steps to convert a database from a non-CDB to a PDB as part of a database upgrade.

Oracle Database 21c requires multitenant architecture. With this release, you can use AutoUpgrade for non-CDB to PDB upgrades from Oracle Database 12c Release 2 (12.2) and later releases. Using the automation features of AutoUpgrade reduces both the effort required of DBAs for upgrades, and the risk associated with typographical errors in a manual process.

Related Topics

- [Oracle® Database Database Upgrade Guide](#)

AutoUpgrade Automates Upgrade of a PDB via Unplug-Plug-Upgrade

AutoUpgrade automates unplugging a PDB from a CDB, plugging it into a higher version CDB, and upgrading the PDB to that new target version.

As part of the process to simplify upgrade, starting with Oracle Database 21c, you can upgrade pluggable databases (PDBs) using an unplug/plug, where the upgrade occurs when you plug the PDB into the target release container database (CDB). This feature adds another scenario in which AutoUpgrade can automate the entire database upgrade process.

Related Topics

- [Oracle® Database Database Upgrade Guide](#)

Oracle Database Automates Database Upgrades with AutoUpgrade

Oracle Database AutoUpgrade enables you to upgrade one or many databases without human intervention, all with one command, and with a single configuration file.

AutoUpgrade enables you to upgrade one or many Oracle Database instances at the command-line, using a single command and a single configuration file. AutoUpgrade runs the preupgrade tasks, performs automated fix-ups where needed, runs the database upgrade, and finishes by completing post-upgrade tasks. It includes automatic retry and fallback, the option to schedule upgrades for future points in time, and the ability to set, change, or remove initialization parameters as desired. Using AutoUpgrade can save you time and money by enabling you to upgrade hundreds of databases with one command, and avoid the need for high-maintenance upgrade solutions.

Related Topics

- [Oracle® Database Database Upgrade Guide](#)

Management Solutions

- [Diagnosability](#)
- [Manageability](#)

Diagnosability

- [Enhanced Diagnosability of Oracle Database](#)
- [SQL*Net Improved Diagnosability](#)

Enhanced Diagnosability of Oracle Database

Diagnosability of database issues is enhanced through a new attention log, as well as classification of information written to database trace files. The new attention log is written in a structured format (XML or JSON) that is much easier to process or interpret and only contains information that requires attention from an administrator. Trace files now contain information that enables easier classification of trace messages.

Enhanced diagnosability features simplify database administration and improve data security.

Related Topics

- [Oracle® Database Administrator's Guide](#)

SQL*Net Improved Diagnosability

Starting with Oracle Database 21c, a connection identifier is available for each network connection. The connection identifier uniquely identifies a connection in trace and logs of different network elements and helps in correlating diagnostic data from these elements.

When a SQL*Net connection has multiple hops, such as from a client to Oracle Connection Manager (CMAN) and then to a server, correlating diagnostic information from the existing logs and traces becomes difficult. However, with the availability of a connection identifier, you can now easily correlate diagnostics, track network data traffic, and resolve connectivity errors.

Related Topics

- [Oracle® Database Net Services Administrator's Guide](#)

Manageability

- Persistent Memory Database
- DAX-Enabled File Systems
- New Database Initialization Parameters for Database Resident Connection Pooling (DRCP)
- Multi-Mount DBFS Client
- Near Zero Brownout for Planned Maintenance
- Oracle Grid Infrastructure SwitchHome
- Read-Only Oracle Home Default

Persistent Memory Database

The Persistent Memory Database (PMEM) support feature enables you to place database files directly on non-volatile memory. The underlying file store is FsDirect, a pointer-switching PMEM file system that supports atomic writes of Oracle Database data blocks. The FsDirect PMEM file store provides the external interface for map and access the Oracle database directly in persistent memory.

Queries can read directly from PMEM without first copying data into the database buffer cache, avoiding data redundancy and unnecessary I/O.

Oracle Persistent Memory Database is ideal for providing high performance with small-scale databases such as those used in Microservices, Sharding, and Read Replicas. Oracle Persistent Memory Database delivers higher transaction rates and fast response-time, achieving unprecedented levels of performance for small-scale systems.

Related Topics

- [Oracle® Database Administrator's Guide](#)

DAX-Enabled File Systems

You must use an XFS-based Data Analytics Accelerator (DAX) file system as the file store for the Persistent Memory (PMEM) database.

You can use a PMEM file store for database data files and control files. For performance reasons, Oracle recommends that you store redo log files as independent files in the file system.

Related Topics

- [Oracle® Database Installation Guide for Linux](#)

New Database Initialization Parameters for Database Resident Connection Pooling (DRCP)

New database initialization parameters, `DRCP_DEDICATED_OPT`, `MIN_AUTH_SERVERS`, and `MAX_AUTH_SERVERS`, have been added to configure Database Resident Connection Pooling (DRCP).

`MIN_AUTH_SERVERS` and `MAX_AUTH_SERVERS` allow the number of processes used to handle session authentication for DRCP to be configured for optimal usage. Authentication server statistics can be viewed in `V$AUTHPOOL_STATS`.

With DRCP, when the number of application connections to the broker is less than the maximum pool size, a "dedicated optimization" makes DRCP behave like dedicated servers. With this optimization, DRCP tends towards a one-to-one correspondence between application connections and DRCP server processes even if those processes are not currently doing database work. Setting `DRCP_DEDICATED_OPT` to `NO` turns off the optimization and reduces the tendency of the pool to grow towards its maximum size until necessary. This helps keep the number of DRCP server processes small when statement execution concurrency is low, therefore reducing memory usage on the database host.

Related Topics

- [Oracle® Database Administrator's Guide](#)

Multi-Mount DBFS Client

DBFS (Database File System) provides a file system interface for storing files/directories in the Oracle database. `dbfs_client`, helps in exposing a DBFS in a database user as a mount point for the OS. The current version of `dbfs_client` can service only the DBFS of a single database user.

Databases can have a number of PDBs each having their own DBFS that they use to store various files (e.g trace files, import dump, user files etc). There can be about 100 PDBs and hence a 100 DBFS to be serviced concurrently. In order to cater to this environment `dbfs_client` needs be able to service multiple DBFS owned by different users across databases. Currently a `dbfs_client` instance can service only one DBFS. Hence, to service 100 DBFS the same number of `dbfs_client` instances would be required. Currently in DBFS, there does not exist a single point of control that can manage all these different `dbfs_client` instances. This could make the management and monitoring of different client instances burdensome for database administrators.

This enhanced version of `dbfs_client` is capable of servicing DBFS of multiple database users. This means that `dbfs_client` can handle multiple mount points, each mount point services DBFS under one database user. This enhanced version of `dbfs_client` is referred to as MUMV (Multi User Mount Version) and the earlier version of `dbfs_client` is referred to as SUMV (Single User Mount Version). The new version of `dbfs_client` can be started in SUMV mode or MUMV mode. If started in SUMV mode, its behavior is the same as the earlier version.

DBFS provides a file system interface for storing files/directories in the Oracle database. Existing single-mount DBFS clients (`dbfs_client`) could mount only one user's DBFS file system. DBFS therefore required multiple DBFS client processes to support multiple file systems on the same host's DBFS file system. This enhanced DBFS multi-mode client can manage different mount points within a single DBFS client process. DBFS client with multi-mount support provides better ease of use and improved performance. Multi-mount DBFS client scales seamlessly to 100s of PDBs and DBFS client can be started as either MUMV (Multi User Mount Version) or SUMV (Single User Mount Version) mode.

Related Topics

- [Oracle® Database SecureFiles and Large Objects Developer's Guide](#)

Near Zero Brownout for Planned Maintenance

Planned maintenance and unplanned outages restart the database instances, but planned maintenance allows for preparation. Near Zero Brownout for Planned Maintenance reduces reconfiguration time for an instance targeted for a planned maintenance operation.

Near Zero Brownout for Planned Maintenance increases the availability of the database during online maintenance operations.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Details: Near Zero Brownout for Planned Maintenance

21c Near Zero Brownout for Planned Maintenance

For planned outages, you can achieve almost no interruption to users (zero brownout) in Oracle Database 21c.

A phased instance shutdown allows surviving instances to take ownership of locks and resources so after it is stopped, the amount of work during reconfiguration is reduced.

When the DBA issues a shutdown command for an instance, the instance is transitioned from an active instance to a passive instance.

When this process is initiated, the following actions take place:

1. Services on the passive instance are stopped and relocated to another instance.
2. After services have been drained from the passive instance, all PDBs are closed.
3. The instance is shutdown normally.

On a Oracle RAC cluster only one passive instance can exist at any time.

For planned outages, you can achieve almost no interruption to users (zero brownout) in Oracle Database 21c. A phased shutdown of an instance allows the surviving instances to take ownership of locks and other resources

so that after the instance is stopped, the amount of work required during reconfiguration is reduced. When the DBA issues a shutdown command for an instance, the instance is transitioned from an active instance to a passive instance.

A passive instance is in an ideal state to be stopped, which means as many resources and locks as possible are transitioned to the active instances. On a Oracle RAC cluster only one passive instance can exist at any time. When this process is initiated, the following actions take place:

1. The services running on the passive instance are stopped and relocated to another instance. This includes moving the connections from the passive instance to the active instance with a targeted drain timeout. The drain timeout indicates how long you want to wait for everything to complete, including the shutting down of the passive instance.
2. After the services have been drained from the passive instance, all pluggable databases (PDBs) are closed. All lock masters are transferred from the passive instance to active instances.
3. The instance is shutdown normally.

Oracle Grid Infrastructure SwitchHome

You can use the `-switchGridHome` option with `gridSetup.sh` to switch from one Oracle Grid Infrastructure home to another.

You can use the `-switchGridHome` option for patching and upgrading Oracle Grid Infrastructure. Use the `-switchGridHome` option to switch from the source Oracle Grid Infrastructure home to the patched Oracle Grid Infrastructure home. All Oracle Clusterware and Oracle Restart services start from the patched Oracle Grid Infrastructure home automatically.

Related Topics

- [Oracle® Grid Infrastructure Grid Infrastructure Installation and Upgrade Guide](#)

Read-Only Oracle Home Default

Read-only Oracle homes, where all configuration data and log files reside outside of the read-only Oracle home, are the default option for Oracle Database installations and upgrades.

Read-only Oracle homes enable an easy, flexible, and software-image based deployment of Oracle software that can automatically and seamlessly be distributed across multiple servers. Read-only Oracle homes also enable patching and updating of Oracle Database without extended downtime, as patching simply means replacing a given set of binaries in a defined location.

Related Topics

- [Oracle® Database Database Installation Guide](#)

Performance and High Availability

- ACFS Cluster File System
- Application Continuity
- Automatic Operations
- Automatic Storage Manager (ASM)
- Autonomous Health Framework
- Clusterware
- Database In-Memory
- Data Guard
- Flashback
- GoldenGate
- Multitenant
- Oracle Real Application Clusters (RAC)
- SecureFiles
- Sharding
- Transactional Event Queues (TEQs)

ACFS Cluster File System

- [AutoShrink for ACFS](#)
- [Mixed Sector Support](#)
- [Oracle ACFS File Based Snapshots](#)
- [Replication Unplanned Failover](#)

AutoShrink for ACFS

Oracle Automatic Storage Management Cluster File System (Oracle ACFS) automatic shrinking automatically shrinks an Oracle ACFS file system based on policy, providing there is enough free storage in the volume.

The benefit is an optimization in performance and space utilization, ensuring that the data migration and associated locking do not cause delays to the running workloads or timeouts.

Related Topics

- [Oracle® Automatic Storage Management Cluster File System Administrator's Guide](#)

Details: Oracle ACFS Automatic Shrinking

This page explains how to implement ACFS Automatic Shrinking using the `acfsutil size` command.

Oracle ACFS Automatic Shrinking

21c ACFS automatic shrinking automatically shrinks an Oracle ACFS file system based on policy, providing there is enough free storage available in the volume.

The automatic shrinking option of `acfsutil size` shrinks a file system by 25% if it is 50% full and it was at least 80% full since the last expansion, or since `mkfs` was run.

ACFS checks once an hour to determine whether the file system meets the automatic shrink criteria

At most one automatic shrink action occurs daily. The operation is run on the file system in the background.

The `-s` option enables automatic shrinking functionality and the `-n` option disables automatic shrinking functionality

```
acfsutil size -s mount_point
```

Oracle ACFS automatic shrinking automatically shrinks an Oracle ACFS file system based on policy, providing there is enough free storage available in the volume.

The automatic shrinking option (`-s`) of the `acfsutil size` command shrinks a file system by 25% if it is 50%

full and it was at least 80% full since the last expansion, or since mkfs was run (if this is a new file system). Oracle ACFS checks once an hour to determine whether the file system meets the automatic shrink criteria. When the criteria has been detected, the automatic shrinking process begins within an hour if automatic shrinking has not already occurred that day. At most one automatic shrink action occurs daily. The operation is run on the file system in the background. If automatic shrinking is enabled, then `acfsutil info fs` displays `AutoShrinkEnabled` in the flags output. Oracle ACFS automatic shrinking is supported on Linux.

Mixed Sector Support

Oracle ACFS mixed sector support enables the Linux primary and accelerator volumes of an Oracle ACFS file system to use a mix of different logical sector sizes, such as 512-bytes and 4096 bytes.

The benefit is flexibility for storage configuration, enabling primary and accelerator volumes to have different logical sector sizes on Linux operating systems.

Related Topics

- [Oracle® Automatic Storage Management Cluster File System Administrator's Guide](#)

Details: Mixed Sector Support

This page details ACFS mixed sector support.

Oracle ACFS Support for Mixed Sector Sizes

21c ACFS mixed sector support enables the Linux primary and accelerator volumes of an ACFS file system to use a mix of different logical sector sizes.

If **COMPATIBLE . ADVN** is set to 20.1 or greater, then primary and accelerator volumes can use a mix of different logical sector sizes, such as 512 bytes and 4096 bytes.

User data IO continues to support transfers as small as 512 bytes for normal user IO requests.

When the ADVN volume of the file system has a logical disk sector size of 4 K, user Direct IO requests should be aligned on 4 K file offsets for the best performance

Oracle ACFS mixed sector support enables the Linux primary and accelerator volumes of an Oracle ACFS file system to use a mix of different logical sector sizes, such as 512 bytes and 4096 bytes.

Oracle ACFS File Based Snapshots

Oracle Automatic Storage Management Cluster File System (Oracle ACFS) file-based snapshots provide the ability to create snapshots of individual Oracle ACFS files in a space efficient manner on Linux.

The benefit is storage efficiency because you only snapshot specific files, not all files in the file system. Example use cases are for virtual machine (VM) image files and PDB snapshot copies.

Related Topics


- [Oracle® Automatic Storage Management Cluster File System Administrator's Guide](#)

Details: Oracle ACFS File Based Snapshots

This page explains how to implement ACFS File Based Snapshots.

Oracle ACFS File Based Snapshots

21c ACFS file based snapshots provides the ability to create snapshots of individual ACFS files in a space efficient manner on Linux

The `acfsutil fshare create` command can be used to create a file in the namespace (fshare) which is shared with the source file. 

The fshare is an exact replica of the source file and functions as a file based snapshot.

This file sharing capability consumes no storage until it is modified and then the storage being modified is copied on write (COW).

The fshare is similar to a file system snapshot except it is created on an individual file basis.

```
acfsutil fshare create source_file_path destination_file_path
```

Oracle ACFS file based snapshots provides the ability to create snapshots of individual Oracle ACFS files in a space efficient manner on Linux.

The `acfsutil fshare create` command can be used to create a file in the namespace (fshare) which is

shared with the source file.

Oracle ACFS File Based Snapshot Examples

The `acfsutil fshare create` command below successfully creates a file share in the current working directory.

```
# acfsutil fshare create myfile1 myfile2
acfsutil fshare create: Fshare operation is complete.
```

The `acfsutil fshare create` command below attempts, and fails, to create a file share of a directory.

```
# acfsutil fshare create dir1 file2 acfsutil
fshare create: Fshare operation did not complete.
acfsutil fshare create: Cannot share a directory.
```

Replication Unplanned Failover

Oracle ACFS replication failover provides unplanned failover where the standby location assumes the role of the primary in case of failure. When a failure occurs, the standby location pursues contact with the primary and in the absence of a response, the standby assumes the primary role, and on recovery of the former primary, the former primary becomes the new standby.

The benefit is faster recovery in the event of unplanned downtime for Oracle ACFS replication.

Related Topics

- [Oracle® Automatic Storage Management Cluster File System Administrator's Guide](#)

Details: Oracle ACFS Replication Unplanned Failover

This page explains how to implement ACFS Replication Unplanned Failover.

Oracle ACFS Replication Unplanned Failover

21c ACFS replication failover provides unplanned failover where the standby location assumes the role of the primary in case of failure.

The `acfsutil repl failover` command reverses the role of a replication standby location so it becomes a replication primary location, or optionally terminates the replication relationship.

The failover command insures that the standby location contains an exact copy of the results of the last successful replication transfer.

If necessary, the command restores the location back to its state as of that transfer.

```
acfsutil repl failover [-T timeout] [snap_shot@]mount_point
```

The example below shows the use of the `acfsutil repl failover` command. The command is invoked on the standby location and specifies the primary location

```
# acfsutil repl failover /repl_data
```

Oracle ACFS replication failover provides unplanned failover where the standby location assumes the role of the primary in case of failure.

The `acfsutil repl failover` command reverses the role of a replication standby location so it becomes a replication primary location, or optionally terminates the replication relationship. The `failover` command insures that the standby location contains an exact copy of the results of the last successful replication transfer. If necessary, the command restores the location back to its state as of that transfer.

The `acfsutil repl failover` command behaves differently based on the scenario in which it was run:

- Both the standby location and corresponding primary location are operating normally.

In this scenario, the command reverses the replication relationship. There is no data loss. Note that `failover` fails in this case if replication is paused. To allow this case to succeed, run the `acfsutil repl resume` command.

- The primary location is not currently available, but you want to wait until it is back online.

In this scenario, the command verifies the status of the replication primary. If the primary is not accessible and the timeout period has expired (if specified), then the command restores the standby location to its state as of the last successful replication transfer and converts it into a replication primary. Some data loss is possible, for example if there was a transfer in process when the primary location became unavailable. When the original primary location becomes available, it is aware that the `failover` command has been run and converts itself into a replication standby location.

- The primary location is not currently available and you do not want to wait until it is back online.

In this scenario, the command verifies the status of the replication primary. If the primary is not accessible and the timeout period has expired (if specified), then the command restores the standby location to its state as of the last successful replication transfer and converts it into a replication primary. Some data loss is possible, for instance if there was a transfer in process when the primary location became unavailable. After the `failover` command has been run, you have two options: A new standby location can be configured using the `acfsutil repl update` command. Alternatively, you can terminate replication by running `acfsutil repl terminate primary` on the new primary.

Application Continuity

- [Application Continuity Protection Check](#)
- [Planned Failover](#)
- [Reset Session State](#)
- [Transparent Application Continuity](#)
- [Transparent Application Continuity in the Oracle Cloud](#)

Application Continuity Protection Check

Application Continuity Protection Check (ACCHK) provides guidance on the level of protection for each application that uses Application Continuity and assists you to increase protection, if required.

ACCHK identifies which application configuration is protected to help you make an informed decision about which configuration to use for maximum protection or how to increase protection level for an application configuration. ACCHK also provides diagnostics for an unsuccessful failover.

Related Topics

- [Oracle® Real Application Clusters Administrator's Guide](#)

Planned Failover

Oracle Database invokes planned failover at points where the database knows that it can replay the session using Application Continuity and that the session is not expected to drain.

Planned failover is an automatic solution that Oracle Database uses for relocating sessions during planned maintenance for batch and long running operations that are not expected to complete in the specified drain timeout period.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Reset Session State

The reset session state feature clears the session state set by the application when the request ends. The `RESET_STATE` database service attribute cleans up dirty sessions, so that the applications, which use these sessions after cleanup, cannot see the state of these sessions.

The `RESET_STATE` feature enables you to clean the session state at the end of each request so that the database developers do not have to clean the session state manually. You can reset session state of the applications that are stateless between requests.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Transparent Application Continuity

In this release, the following new features are provided for Transparent Application Continuity:

- Oracle Database clients use implicit request boundaries when connecting to the database using a service that has the attribute `FAILOVER_TYPE` set to `AUTO`.
- Planned Failover is introduced, which is failover that is forced by the Oracle database at points where the Oracle database decides that the session can be failed over and the session is unlikely to drain. This feature is also available for Application Continuity.
- There is also a new service attribute, `RESET_STATE`. The resetting of state is an important feature that clears the session state set by the application in a request at the end of request. This feature is also available for Application Continuity.

This improvement increases coverage of Transparent Application Continuity for applications that do not use an Oracle-supplied connection pool. Planned failover is used for shedding sessions during planned maintenance. It is also used for load rebalancing. Without `RESET_STATE`, application developers need to cancel their cursors, and clear any session state that has been set.

Related Topics

- [Oracle® Database Development Guide](#)

Transparent Application Continuity in the Oracle Cloud

Transparent Application Continuity is enabled by default in an Oracle Cloud environment. Enabling Transparent Application Continuity by default in the Oracle Cloud improves runtime performance, planned failover, and provides broader application coverage.

Transparent Application Continuity reduces overhead, reduces resource consumption, and broadens replay capabilities, so that database requests can replay in Oracle Cloud. Transparent Application Continuity also ensures Continuous Availability for applications working with databases in the Oracle Cloud.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Automatic Operations

- Automatic Indexing Enhancements
- Automatic Index Optimization
- Automatic Materialized Views
- Automatic SQL Tuning Set
- Automatic Temporary Tablespace Shrink
- Automatic Undo Tablespace Shrink
- Automatic Zone Maps
- Object Activity Tracking System
- Sequence Dynamic Cache Resizing

Automatic Indexing Enhancements

Automatic indexing considers more cases for potential indexes and allows inclusion or exclusion of specific tables. An enhancement has been introduced to reduce the overhead of cursor invalidations when a new automatic index is created.

The enhancements increase the number cases where automatic indexes improve query performance.

Related Topics

- [Oracle® Database Administrator's Guide](#)

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. The 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 also consume 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](#)

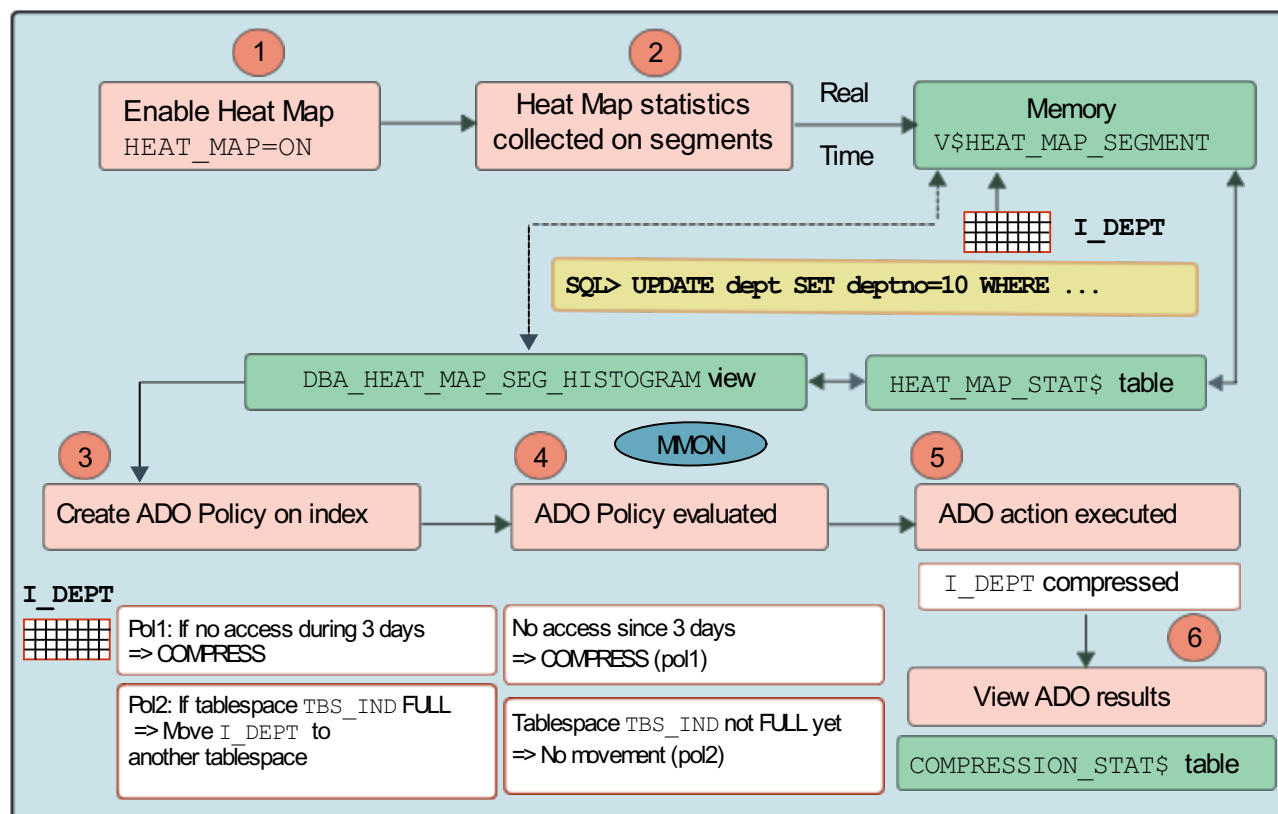
This practice shows how to automate the movement of indexes to another tablespace depending on certain conditions defined in Automatic Data Optimization policies.

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 21c 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 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

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1 : Set up the environment for testing

- Execute the shell script that 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
Copyright (c) 1982, 2019, Oracle. All rights reserved.

Connected to:

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

214 rows created.

SQL> COMMIT;

Commit complete.

SQL> exit
$
```

Step 2 : Display the space used and freed by the table index in the tablespace

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


```

$ sqlplus system@PDB21

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

Enter password:

Connected to:

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_n
ame='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>

```

- Display the space used and free in the tablespace in 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	310	11.625	4	96
SYSAUX	450	25.4375	6	94
USERS	671.25	37.8125	6	94
TBS_FOR_ADO	4	1.3125	33	67
ADOTBSINDX	2	.9375	47	53
UNDOTBS1	250	210.75	84	16
LOW_COST_STORE_INDX	100	99	99	1

7 rows selected.

SQL>

Step 3 : Create a storage tiering ADO policy on the index

- 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>

- Display the policy in the data dictionary view.

```
SQL> CONNECT hr@PDB21
Enter 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
P2	STORAGE	SEGMENT	LOW_COST_STORE_INDX

```
SQL>
```

Step 4 : Test the storage tiering ADO policy

- Insert rows into HR.EMP until the index entries inserted raise the percentage of empty space in ADOTBSIN DX 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	310	11.625	4	96
SYSAUX	450	25.4375	6	94
USERS	671.25	37.8125	6	94
TBS_FOR_ADO	4	1.25	31	69
ADOTBSINDX	2	.875	44	56
UNDOTBS1	250	210.75	84	16
LOW_COST_STORE_INDX	100	99	99	1

7 rows selected.

```
SQL>
```

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

- Display the tablespace in which the index of the primary key for the HR.EMP table is now stored. Did the index move 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%.

- The ADO decision to move segments also depends on the default thresholds defined at the database level for all user-defined tablespaces.
 - Set the TBS_PERCENT_FREE threshold to 90% and the TBS_PERCENT_USED threshold to 30% .

```

SQL> CONNECT sys@PDB21 AS SYSDBA
Enter 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,9
0)

PL/SQL procedure successfully completed.

SQL> EXEC dbms_ilm_admin.customize_ilm(DBMS_ILM_ADMIN.TBS_PERCENT_USED,3
0)

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>

```

- o Also, specify that seconds (rather than days) should be used, to test ADO policy evaluation quickly instead of waiting for the policy duration.

```

SQL> EXEC dbms_ilm_admin.customize_ilm(dbms_ilm_admin.POLICY_TIME,dbms_i
lm_admin.ILM_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                      30
TBS PERCENT FREE                      90
POLICY TIME                            1

8 rows selected.

SQL>

```

- For the purpose of the practice, you will not wait for the maintenance window to open to trigger the ADO policies jobs. Instead, you are going to execute the following commands and PL/SQL block, connected as the ADO policy owner HR.

```

SQL> CONNECT hr@PDB21
Enter 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_of
fline,
                        task_id    => v_executionid);
END;
/

PL/SQL procedure successfully completed.

SQL>

```

- 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      ILMJOB100

SQL> SELECT task_id, job_name, job_state FROM user_ilmresults;

TASK_ID JOB_NAME      JOB_STATE
-----
       1 ILMJOB100 COMPLETED SUCCESSFULLY

SQL>
```

- Display the tablespace in which the index of the primary key for the `HR.EMP` table is now stored. Has it moved to the `LOW_COST_STORE_INDX` tablespace?

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

TABLESPACE_NAME      INDEX_NAME      OWNER
-----
LOW_COST_STORE_INDX  PK_EMPLOYEE_ID  HR

SQL>
```

The index has moved to the other tablespace.

Step 5 : Drop the ADO policy

- Delete the ADO policy on the index.


```
SQL> CONNECT system@PDB21
Enter password:
Connected.
SQL> ALTER INDEX hr.pk_employee_id ILM DELETE POLICY p2;

Index altered.

SQL>
```

- 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 Materialized Views

Materialized views offer the potential to improve query performance significantly, but considerable effort and skill is required to identify what materialized views to use. The database now incorporates workload monitoring to establish what materialized views are needed. Based on the decisions it makes, materialized views and materialized view logs are created and maintained automatically without any manual interaction.

Automatic materialized views improve application performance transparently without the need for any user action or management overhead.

Related Topics

- [Oracle® Database Performance Tuning Guide](#)

Automatic SQL Tuning Set

The automatic SQL tuning set (ASTS) is a system-maintained record of SQL execution plans and SQL statement performance metrics seen by the database. Over time, the ASTS will include examples of all queries seen on the system.

SQL plans and metrics stored in ASTS are useful for repairing SQL performance regressions quickly using SQL plan management.

ASTS enables SQL performance regressions to be resolved quickly and with minimal manual intervention. It is complementary to the automatic workload repository (AWR) and considered a similar core manageability infrastructure of the Oracle Database.

Related Topics

- [Oracle® Database SQL Tuning Guide](#)

Automatic Temporary Tablespace Shrink

A temporary tablespace can grow to a very large size due to spikes in temp usage by queries. Sorts, hash joins, and query transformations are examples that might cause high temp usage. Automatic Temp Tablespace Sizing is a feature which shrinks the temporary tablespace in the background when temp usage has subsided. In addition, if temp usage is increasing, the feature would pre-emptively grow the temporary tablespace to ensure query performance is not impacted.

This feature alleviates the need for a DBA to manually size the temporary tablespace. These are the two scenarios which would benefit:

- The DBA does not need to manually shrink the temporary tablespace to reclaim unused space.
- The DBA does not need to manually grow the temporary tablespace in anticipation of high temp usage.

Automatic Undo Tablespace Shrink

An undo tablespace can grow to a very large size and then that space may not be needed again. Currently there is no automated way to recover that space and there have been limits placed on the size of the undo tablespace in some environments because space cannot be easily reclaimed. This can prevent large transactions from running successfully. Automatic Undo Tablespace Shrink is a feature that shrinks the undo tablespace in the background by dropping expired (i.e. manual or automatically configured) undo segments and their corresponding extents, and then performing a datafile shrink if possible. A datafile shrink is not guaranteed depending on where allocated extents are located. Even if datafile shrink is not possible, releasing undo extents back to the undo tablespace enables the reuse of space by other undo segments.

This feature reduces space requirements and the need for manual intervention:

- Automatically recovering space used by transactions that are no longer active
- Remove the need to restrict the size of the undo tablespace allowing larger transactions to run with the ability to recover that space once the transaction undo expires.

Automatic Zone Maps

Automatic zone maps are created and maintained for any user table without any customer intervention. Zone maps allow the pruning of block ranges and partitions based on the predicates in the queries. 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 improve the performance of queries transparently and automatically without management overhead.

[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.

Related Topics

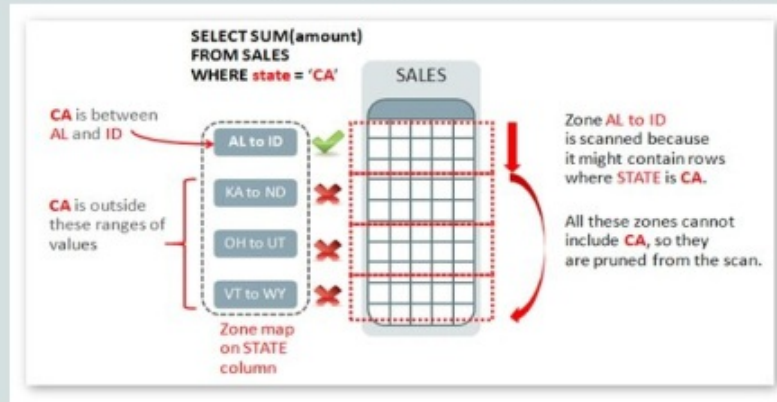
- [Oracle® Database Data Warehousing Guide](#)

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.

21c

- **Automatic** zone maps are created and maintained for any user table **without any customer intervention**. Zone maps can still be created manually.

Oracle Database 21c 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 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.

DBMS_AUTO_ZONEMAP package

Configuration

- Disable auto zone map: `DBMS_AUTO_ZONEMAP.CONFIGURE ('AUTO_ZONEMAP_MODE', 'OFF')`

- Enable auto zone map:

`DBMS_AUTO_ZONEMAP.CONFIGURE ('AUTO_ZONEMAP_MODE', 'ON')`

`DBMS_AUTO_ZONEMAP.CONFIGURE ('AUTO_ZONEMAP_MODE', 'BACKGROUND')`

`DBMS_AUTO_ZONEMAP.CONFIGURE ('AUTO_ZONEMAP_MODE', 'FOREGROUND')`

Reporting

Reports automatic zone maps activity for a given time window:

- All the activity of the last execution:

`SELECT DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT () FROM dual;`

- All the activity of the last 2 days:

`SELECT DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT (SYSTIMESTAMP-2, NULL) FROM dual;`

- All the activity with all sections with typical details for the last 48 hours in text format:

`DBMS_AUTO_ZONEMAP.ACTIVITY_REPORT (SYSTIMESTAMP-2, SYSTIMESTAMP, 'TEXT', 'ALL', 'TYPICAL')`

The new `DBMS_AUTO_ZONEMAP.CONFIGURE` package and procedure allow you to set configuration options for automatic zone maps, 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` procedure reports automatic zone maps activity for a given time window. Because the zone maps autotask background job is scheduled for every 15 minutes and runs 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 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 within each section. Possible values are BASIC, TYPICAL and ALL. Default is TYPICAL.
 - **BASIC:** Presents very high level details in an executive summary. Users only see numbers on zone maps that were created, complete rebuilt and fast rebuilt. In the new zone map details section, you can see the 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 more comprehensive than basic. This level shows full overview in the 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 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. Shows 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.

New Views

DBA_ZONEMAP_AUTO_ACTIONS

Get important insights such as:

- How many zone maps were created across all executions?
- How many fully stale zone maps were rebuilt across all executions?
- How many partial stale zone maps were rebuilt across all executions?

DBA_ZONEMAP_AUTO_FINDINGS

Get important findings such as:

- Get all evicted base tables during zone map creation
- Get all base tables which had errors during zone map creation

Reports automatic zone maps activity for a given time window:

Existing View

DBA_ZONEMAPS

Three new columns added:

- Is the zone map automatically created or not?
- Is the zone map partly stale or not?
- Is the zone map missing zones or not?

The `DBA_ZONEMAP_AUTO_ACTIONS` view contains 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` view contains 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 time out.
- `FINDING_TYPE`: Finding type can be a blacklist, back in queue and others.

The existing `DBA_ZONEMAPS` view contains 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?

Object Activity Tracking System

Object Activity Tracking System (OATS) tracks the usage of various types of database objects. Usage includes operations such as access, data manipulation, or refresh.

Automated tracking of how database objects are being used enables customers to gain a better insight into how applications are querying and manipulating the database and its objects. Internal clients such as Access Advisors or Automatic Materialized Views leverage and benefit from OATS as well.

Related Topics

- [Oracle® Database Performance Tuning Guide](#)

Sequence Dynamic Cache Resizing

With dynamic cache resizing, the sequence cache size is now auto-tuned based on the rate of consumption of sequence values. This means the cache can automatically grow and shrink over time, depending on usage, while never falling below the DDL specified cache size. By dynamically resizing the cache, performance can be improved significantly, especially for fast insert workloads on Oracle RAC, by reducing the number of trips to disk needed to replenish the cache.

Dynamic cache resizing can improve performance significantly for fast insert workloads that use sequences. This is accomplished by reducing the number of trips to disk needed to replenish the sequence cache and can be especially significant in an Oracle RAC environment.

Related Topics

- [Oracle® Database Database Administrator's Guide](#)

Automatic Storage Manager (ASM)

- [Enable ASMCA to Configure Flex ASM on an Existing NAS Configuration](#)
- [Enhanced Double Parity Protection for Flex and Extended Disk Groups](#)
- [File Group Templates](#)
- [Oracle ASM Flex Disk Group Support for Cloning a PDB in One CDB to a New PDB in a Different CDB](#)

Enable ASMCA to Configure Flex ASM on an Existing NAS Configuration

This feature enables you to install Oracle Flex ASM on a configuration previously configured on network file storage (NFS). In particular, Oracle ASM Configuration Assistant (ASMCA) can be run in silent mode to configure Oracle ASM after an Oracle Clusterware installation has been performed using network attached storage (NAS) for Oracle Cluster Registry (OCR) and Voting disks.

The business value of this feature is that it provides an easy way for you to transition from NFS storage over to Oracle ASM managed storage. Without this feature, you would have to do a complete fresh installation and move all databases.

Related Topics

- [Oracle® Automatic Storage Management Administrator's Guide](#)

Enhanced Double Parity Protection for Flex and Extended Disk Groups

This feature provides support for double parity protection for write-once files in an Oracle Automatic Storage Management (Oracle ASM) Flex Disk Group.

With this feature you can use double parity protection for write-once files in a Oracle ASM Flex Disk Group. Double parity protection provides greater protection against multiple hardware failures. A previous release of Oracle ASM provided for simple parity protection for write-once files in a Flex Disk Group. Write-once files include files such as database backup sets and archive logs. The benefit of parity protection as compared to conventional mirroring is that it reduces storage overhead, but with a slight increase of risk of data loss after an event involving multiple hardware failures.

Related Topics

- [Oracle® Automatic Storage Management Administrator's Guide](#)

File Group Templates

With file group templates you can customize and set default file group properties for automatically created file groups, enabling you to customize file group properties that are inherited by a number of databases.

Without file group templates, if you wanted to change properties for an automatically created file group, you would have to manually change the properties after the associated files are created which triggers an unnecessary rebalance. The file group templates feature provides a much better option.

Related Topics

- [Oracle® Automatic Storage Management Administrator's Guide](#)

Oracle ASM Flex Disk Group Support for Cloning a PDB in One CDB to a New PDB in a Different CDB

Previously point-in-time database clones could only clone a pluggable database (PDB) in a multitenant container database (CDB) to a new PDB in the same CDB. The latter restriction is removed as part of this feature. Now, you can clone a PDB in a CDB to a new PDB in a different CDB.

This feature enables you to use Oracle ASM cloning for test and development cloning where the cloned PDB must be in a separate CDB.

Related Topics

- [Oracle® Automatic Storage Management Administrator's Guide](#)

Autonomous Health Framework

- Enhanced Support for Oracle Exadata
- Oracle Cluster Health Advisor Support for Solaris
- Oracle Cluster Health Monitor Local Mode Support
- Oracle ORAchk and EXAchk Support for REST API
- Oracle Trace File Analyzer Real-Time Health Summary
- Oracle Trace File Analyzer Support for Efficient Multiple Service Request Data Collections
- Remote GIMR Support for Oracle Standalone Clusters
- Support for Automatically Enabling Oracle Database Quality of Service (QoS) Management
- Support for Deploying Grid Infrastructure Management Repository (GIMR) into a Separate Oracle Home

Enhanced Support for Oracle Exadata

The ability of Oracle Cluster Health Advisor to detect performance and availability issues on Oracle Exadata systems has been improved in this release with the addition of Exadata specific models.

Oracle Cluster Health Advisor detects performance and availability issues using Oracle Database and node models that were developed using machine learning.

With the improved detection of performance and availability issues on Oracle Exadata systems, Oracle Cluster Health Advisor helps to improve Oracle Database availability and performance. New Exadata-specific models are automatically loaded when CHA runs on Exadata Engineered Systems.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

Oracle Cluster Health Advisor Support for Solaris

Oracle Cluster Health Advisor supports Oracle Real Application Clusters (Oracle RAC) deployments on Oracle Solaris.

With the Oracle Cluster Health Advisor support for Oracle Solaris, you can now get early detection and prevention of performance and availability issues in your Oracle RAC database deployments.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

Oracle Cluster Health Monitor Local Mode Support

You can now configure Oracle Cluster Health Monitor to operate in local mode to report the operating system metrics using the `oclumon dumpnodeview` command even if you have not deployed Grid Infrastructure Management Repository (GIMR).

In local mode, you can get only the local node data. The local mode has limited Oracle Cluster Health Monitor functionality in the deployments where you have not installed Grid Infrastructure Management Repository (GIMR). In earlier releases, Oracle Cluster Health Monitor required GIMR to report the operating system metrics using the `oclumon dumpnodeview` command.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

Oracle ORAchk and EXAchk Support for REST API

In this release, support for REST API adds a remote interface to the existing Oracle ORAchk and Oracle EXAchk command-line interfaces (CLI).

You can manage Oracle software deployments remotely from centralized consoles and web interfaces. By supporting the REST interfaces, Oracle ORAchk and Oracle EXAchk integrate into these applications and help support fleet or cloud management.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

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 Support for Efficient Multiple Service Request Data Collections

Oracle Trace File Analyzer collects multiple Service Request Data Collections into a single collection even if it detects multiple issues or errors at the same time.

Service Request Data Collection mode of operation enables you to collect only the log and trace files that are required for diagnosing a specific type of problem. Even with this optimization, Oracle Trace File Analyzer collects the same subset of files if it detects multiple issues or errors at the same time. The enhancement further optimizes the collection of multiple Service Request Data Collections into a single collection and thus removes duplication.

It is essential to collect log and trace files upon detection of issues before the files are rotated or purged. However, collecting log and trace files involves resource overhead, which may be critically low due to these issues. The enhancement in this release reduces the resource overhead and disk space needed at a critical time.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

Remote GIMR Support for Oracle Standalone Clusters

The remote Grid Infrastructure Management Repository (GIMR) feature for Oracle Standalone Cluster enables you to use a centralized GIMR. This feature does not require local cluster resources to host the GIMR.

The remote GIMR feature provides access to a persistent data store that significantly enhances the proactive diagnostic functionality of Cluster Health Monitor, Cluster Health Advisor, and Autonomous Health Framework clients. The remote GIMR feature saves cost by freeing up local resources and licensed database server resources.

Related Topics

- [Oracle® Grid Infrastructure Grid Infrastructure Installation and Upgrade Guide](#)

Support for Automatically Enabling Oracle Database Quality of Service (QoS) Management

Oracle Database Quality of Service (QoS) Management automatically configures a default policy set based upon the services it discovers and begins monitoring in measurement mode.

With this implementation, the workload performance data is always available to you and other Oracle Autonomous Health Framework components.

If you do not have Oracle Enterprise Manager deployed to monitor Oracle Database clusters, then you cannot utilize the functionality of Oracle Database QoS Management because you cannot enable it with Enterprise Manager. With automatic monitoring, you can now take advantage of the rich set of workload data provided.

In conjunction with the new REST APIs, you can integrate the advanced Oracle Database QoS Management modes into your management systems. In earlier releases, you have to configure the monitoring functionality of Oracle Database QoS Management and enable Oracle Database QoS Management with Enterprise Manager.

Related Topics

- [Oracle® Autonomous Health Framework User's Guide](#)

Support for Deploying Grid Infrastructure Management Repository (GIMR) into a Separate Oracle Home

Starting with this release of Oracle Grid Infrastructure, you must configure the Grid Infrastructure Management Repository (GIMR) in a separate Oracle home, instead of in the Grid home. This option is available when you configure GIMR during a fresh Oracle Grid Infrastructure installation or you add a GIMR to an existing deployment. It is mandatory to configure GIMR in a separate Oracle home when you upgrade Oracle Grid infrastructure with an existing GIMR deployed in it.

A separate Oracle home for the GIMR ensures faster rolling upgrades, fewer errors, and fewer rollback situations. The Oracle Grid Infrastructure installation owner user must own the GIMR home.

Related Topics

- [Oracle® Grid Infrastructure Grid Infrastructure Installation and Upgrade Guide](#)

Clusterware

- [Clusterware REST API](#)
- [Common Data Model Across Fleet Patching and Provisioning Servers](#)
- [FPP Integration with AutoUpgrade](#)
- [Details: Oracle Clusterware 21c Deprecated and Desupported Features](#)

Clusterware REST API

The Clusterware REST APIs enable customers to remotely execute commands on their cluster and to monitor the execution, including output, error codes, and time to execute. Support is provided for existing Oracle Clusterware command line interfaces.

REST API-based management based on well-known Oracle Clusterware command line interfaces simplifies cluster management in the Oracle Cloud, at remote physical locations or locally provisioned.

Related Topics

- [Oracle® Database REST APIs for Oracle Database](#)

Common Data Model Across Fleet Patching and Provisioning Servers

The common data model across Fleet Patching and Provisioning (FPP) servers provides a unified view of fleet targets regardless of the FPP server deployment.

The unified view of the common model provides an easier and simplified operation of large estates and cloud management across multiple data centers.

Related Topics

- [Oracle® Clusterware Administration and Deployment Guide](#)

Details: Fleet Patching and Provisioning Common Data Model

This slide explains how existing RHP peer server functionality has been extended to support query operations, achieving a globally shared data model among the servers.

21c Fleet Patching and Provisioning Common Data Model

FPP provides support for peer servers allowing the management of images between the peers.

This has been extended to enable querying the information for the rest of the entities stored in the repository of peer servers.

The `-rhpserver` option has been added to query operations:

- `rhpcctl query client`
- `rhpcctl query image`
- `rhpcctl query imagetype`
- `rhpcctl query series`
- `rhpcctl query server`
- `rhpcctl query workingcopy`

The command below lists the details for image image1 from all servers:

```
$ rhpcctl query image -image image1 -rhpserver .+
```

FPP provides support for peer servers, allowing the management of images between the peers. This functionality has been extended to allow querying the information for the rest of the entities stored in the repository of peer servers.

The `-rhpserver` option is available for query operations on RHP servers only. The argument passed from the command line to the option is treated as a Java regular expression. For each site that is either the local RHP server or a peer server and whose name matches the expression provided, the query will be executed on it. The `-rhpserver` option has been added to query operations:

- `rhpcctl query client`
- `rhpcctl query image`
- `rhpcctl query imagetype`
- `rhpcctl query series`
- `rhpcctl query server`
- `rhpcctl query workingcopy`

FPP Integration with AutoUpgrade

Fleet Patching and Provisioning (FPP) integration with AutoUpgrade provides a new tool for automating and simplifying Oracle Database Upgrade.

This feature makes Oracle Database AutoUpgrade more flexible, provides better control over the upgrade flow mechanism, and provides better usability by showing progress bar and additional elements. You can upgrade multiple databases in parallel.

Related Topics

- [Oracle® Fleet Patching and Provisioning Administrator's Guide](#)

Details: Oracle Clusterware 21c Deprecated and Desupported Features

This slide lists deprecated and desupported features in Oracle Clusterware 21c.

21c Deprecated and Desupported Features in Oracle Clusterware 21c

- Deprecation of Policy-Managed Databases.
- Deprecation of Cluster Domain - Domain Services Cluster
- Deprecation of Policy-Managed Databases.
- Vendor Clusterware Integration with Oracle Clusterware has been desupported.
- Desupport of Cluster Domain - Member Clusters

Starting with Oracle Grid Infrastructure 21c, creation of new server pools is eliminated, and policy-managed databases are deprecated and can be desupported in a future release. Server pools will be migrated to the new Oracle Services feature that provides similar functionality.

With the introduction of Oracle Grid Infrastructure 21c, Domain Services Cluster (DSC), which is part of the Oracle Cluster Domain architecture, are deprecated and can be desupported in a future release.

Starting with Oracle Oracle Clusterware 21c , the integration of vendor or third party clusterware with Oracle Clusterware is desupported.

Member Clusters, which are part of the Oracle Cluster Domain architecture, are desupported. However, Domain Services Clusters continue to support Members Clusters in releases previous to Oracle Grid Infrastructure 21c.

Database In-Memory

- [Database In-Memory Base Level](#)
- [Automatic In-Memory](#)
- [Database In-Memory External Table Enhancements](#)
- [In-Memory Hybrid Scans](#)
- [CellMemory Level](#)
- [Database In-Memory: Additional Features](#)

In addition to the features highlighted in the Database In-Memory section, the following features in other sections are applicable for Database In-Memory:

- [In-Memory Full Text Columns](#)
- [Spatial Support for Database In-Memory](#)
- [In-Memory Deep Vectorization](#)
- [New JSON Data Type](#)

Database In-Memory Base Level

Database In-Memory is an option to Enterprise Edition and now has a new "Base Level" feature. This allows the use of Database In-Memory with up to a 16GB column store without having to license the option. The use of the Base Level features does not trigger any license tracking.

The IM column store is limited to 16GB when using the Base Level feature. This can allow customers to see the value of Database In-Memory without having to worry about licensing issues. Note that Base Level has some other restrictions; for instance, the CellMemory feature and Automatic In-Memory are not included with the base level.

Related Topics

- [Oracle® Database In-Memory Guide](#)

Automatic In-Memory

Automatic In-Memory (AIM) enables, populates, evicts, and recompresses segments without user intervention.

When `INMEMORY_AUTOMATIC_LEVEL` is set to `HIGH` the database automatically populates segments based on their usage patterns without requiring them to be marked `INMEMORY`. Combined with support for selective column level 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 (IM column store) at one time and maximizes overall application performance.

[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 `INMEMORY_AUTOMATIC_LEVEL` initialization parameter 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 `INMEMORY_AUTOMATIC_LEVEL` initialization parameter 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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1 : Set up the environment with In-Memory Column Store

The shell script configures the IM column store to 110M, creates `NO INMEMORY` tables in HR schema in PDB21, and finally inserts rows in HR tables.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/AutoIM_setup.sh
...
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> CREATE TABLESPACE imtbs DATAFILE SIZE 10G;

Tablespace created.

SQL> EXIT
...
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> /

1753088 rows created.

SQL> COMMIT;

Commit complete.

SQL> EXIT
$
```

Step 2: Configure in-memory tables

- Query the data dictionary to determine whether HR tables are specified as INMEMORY.

```
$ sqlplus sys@PDB21 AS SYSDBA
Enter password:

Connected to:

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>
```

- 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>
```

Step 3 : Configure Automatic In-Memory

- 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> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL>
```

- Query the data dictionary to determine whether HR tables are specified as INMEMORY.

```

SQL> CONNECT sys@PDB21 AS SYSDBA
Enter 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>

```

- Set INMEMORY_AUTOMATIC_LEVEL to HIGH at the PDB level, and re-start PDB21.

```
SQL> ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL=HIGH SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE
Pluggable Database closed.
SQL> STARTUP
Pluggable Database opened.
SQL>
```

Step 4 : Test

- 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>
```

- Populate the in-memory tables 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.constra
int_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> /
```

27392 rows created.

```
SQL> COMMIT;
```

Commit complete.

```
SQL> /
```

...

```
SQL> /
```

Commit complete.

```
SQL> COMMIT;
```

Commit complete.

```
SQL>
```

Why aren't the `ENABLED AUTO` tables 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.

- Execute the `/home/oracle/labs/M104783GC10/AutoIM_scan_AUTO.sql` SQL script to insert more rows into the `HR.EMPLOYEES` table, query the `HR.EMPLOYEES` table, and then possibly get the table automatically populated 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>
```

- 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> COL segment_name FORMAT A18
SQL> SELECT segment_name, inmemory_size, bytes_not_populated, inmemory_compression FROM v$im_segments;
```

SEGMENT_NAME	INMEMORY_SIZE	BYTES_NOT_POPULATED	INMEMORY_COMPRESS
PROMOTIONS	1310720		0 AUTO
SALES	1310720		0 AUTO
SALES	1310720		0 AUTO
TIMES	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
SALES_ZM	35717120		0 AUTO
SALES	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
SALES	1310720		0 AUTO
COSTS	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
EMP	24969216	125337600	FOR QUERY LOW
SALES	1310720		0 AUTO
COSTS	1310720		0 AUTO
SALES	1310720		0 AUTO
SALES	1310720		0 AUTO

23 rows selected.

```
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 table may be evicted according to the internal statistics. If you re-query the HR.EMP table, the server may decide to evict the HR.EMPLOYEES table to let HR.EMP back into the IM column store.

Database In-Memory External Table Enhancements

For a partitioned external table or a hybrid partitioned table (a table which has both internal and external partitions), the `INMEMORY` clause is supported at both the table and partition level. For hybrid partitioned tables, the table-level `INMEMORY` attribute applies to all partitions, whether internal or external.

This enhancement significantly broadens support for in-memory external tables by allowing partitioned external tables or hybrid partitioned tables to be selectively populated by partition thereby supporting active subsets of external data rather than having to populate the entire table.

Related Topics

- [Oracle® Database In-Memory Guide](#)

In-Memory Hybrid Scans

In-Memory hybrid scans support 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 previous releases if a column that was not populated was accessed by a query then no data could be accessed from the IM column store.

The In-Memory hybrid scan feature allows a query to combine in-memory accesses with row-store accesses, 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 21c.

[Practice: Using In-Memory Hybrid Scans in Queries](#)

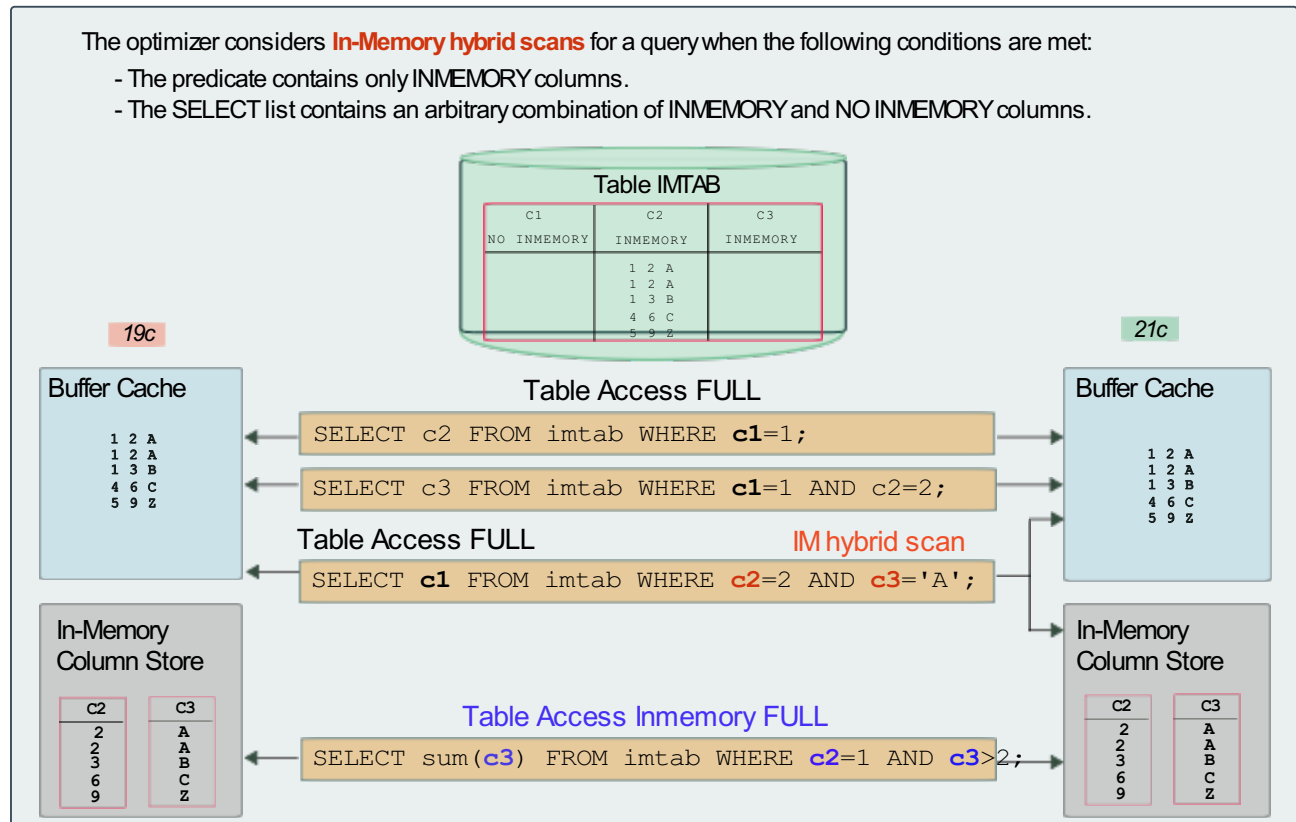
This practice shows how queries referencing both `INMEMORY` and `NO INMEMORY` columns can access columnar data. This optimizer access method called IM hybrid scan can improve performance by orders of magnitude. If the optimizer chooses a table scan, the storage engine automatically determines whether 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 21c.



Before Oracle Database 21c, 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 21c, 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

Overview

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.

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment with In-Memory Column Store

The `IM_Hybrid_setup.sh` shell script configures the IM column store to 110M, creates an in-memory table named `IMU.IMTAB` containing two `INMEMORY` columns and one `NO INMEMORY` column, and populates the table. The shell script executes the same operations in Oracle Database 19c and Oracle Database 21c.

- Run the `IM_Hybrid_setup.sh` script.

```
$ cd /home/oracle/labs/M104783GC10
$ /home/oracle/labs/M104783GC10/IM_Hybrid_setup.sh
...
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.

...
SQL> CREATE TABLESPACE imtbs DATAFILE 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_noi
nmem);

Table created.

SQL> INSERT INTO imu.imtab VALUES (3,4,'Test21c');

1 row created.

SQL> INSERT INTO imu.imtab SELECT c1_noinmem + (select max(c1_noinmem) from imu.im
tab),
  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.
...
131072 rows created.

SQL> COMMIT;

Commit complete.

SQL> exit
$
```

Step 2 : Populate the in-memory table

- Connect to PDB21 as SYSTEM and set formats for the queried columns.


```
SQL> SELECT /*+ FULL(imu.imtab) NO_PARALLEL(imu.imtab) */ COUNT(*) FROM imu.i
mtab;

COUNT (*)
-----
262144

SQL>
```

- 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>
```

Step 3 : Complete In-Memory Scans

- Execute a 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 chose the `TABLE ACCESS FULL` method because the predicate does not contain only `INMEMORY` columns.

- 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 'Test21c%';
```

```
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 'Test21c%'
```

```
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
'Test21c%'))
```

```
Note
```

```
-----
```

```
- dynamic statistics used: dynamic sampling (level=2)
```

```
25 rows selected.
```

```
SQL>
```

The optimizer in both sessions chose the TABLE ACCESS FULL access method because the predicate does not contain only INMEMORY columns. It contains an INMEMORY column and a NO INMEMORY columns.

- 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 'Test21c%';
```

```
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 'Test21c%'
```

```
Plan hash value: 360700294
```

```
-----
-----
| Id | Operation | Name | Rows | Bytes | Cost ( |
|CPU)| Time | | | | %CPU) |
-----
-----
| 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 'Test21c%'))
```

```
Note
```

```
-----
- dynamic statistics used: dynamic sampling (level=2)
```

```
24 rows selected.
```

```
SQL>
```

The optimizer in both sessions chose different access methods. In 21c, 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.

Step 4 : Drop the user

- Drop the `imu` user.

```
SQL> DROP USER imu CASCADE;
```

```
User dropped.
```

```
SQL> EXIT
```

```
$
```

CellMemory Level

On Exadata systems you can use both memory in the compute servers for the In-Memory column store and the Exadata Smart Flash Cache in the storage servers to populate data in in-memory columnar format (a feature known as *CellMemory*). CellMemory is enabled by default if the IM column store is enabled on the database servers. You can now selectively enable only CellMemory without enabling the IM column store, by setting `INMEMORY_FORCE=CELLMEMORY_LEVEL` and `INMEMORY_SIZE=0`.

If you do not intend to use Database In-Memory on the Database servers, this feature allows you to use CellMemory without incurring the overhead of allocating SGA memory for the IM column store.

Related Topics

- [Oracle® Database In-Memory Guide](#)

Database In-Memory: Additional Features

In addition to the features highlighted in the Database In-Memory section, the following features in other sections are applicable for Database In-Memory:

- [In-Memory Full Text Columns](#)
- [Spatial Support for Database In-Memory](#)
- [In-Memory Vectorized Joins](#)
- [New JSON Data Type](#)

Data Guard

- Active Data Guard - Standby Result Cache
- Data Guard Broker Far Sync Instance Creation
- Data Guard Far Sync instance in Maximum Performance Mode
- Fast-Start Failover Callouts
- Fast-Start Failover Configuration Validation
- PDB Recovery Isolation
- Standardized Data Guard Broker Directory Structure
- Other features

Active Data Guard - Standby Result Cache

The result cache in an Active Data Guard standby database is utilized to cache results of queries that were run on the physical standby database. In the case of a role transition to primary, the standby database result cache will now be preserved ensuring performance for offloaded reporting and other queries continue without compromising the performance benefits of the standby result cache.

Use of the result cache greatly improves query performance for recurring queries and minimizes performance impact on the primary and standby databases. By maintaining the result cache on the standby, the performance of any queries that were running on the standby will be maintained ensuring previously offloaded reporting and other read-only applications utilizing the standby will not be impacted by the role transition.

Related Topics

- [Oracle® Data Guard Concepts and Administration](#)

Data Guard Broker Far Sync Instance Creation

The Data Guard Broker now enables users to create and add a Far Sync instance to a Data Guard Broker configuration using a single command.

Zero Data loss over long distance can be achieved by using the Data Guard Far Sync standby instances. To ease the setup and maintenance of these instances, the Oracle Data Guard Broker can now be utilized. This leads to easier and simplified setup, which leverages the existing maintenance solution of the overall Data Guard environment.

Related Topics

- [Oracle® Data Guard Broker](#)

Details: Data Guard Broker Far Sync Instance Creation

This slide explains how to add a far sync instance to a Data Guard Broker configuration using the `CREATE FAR_SYNC` command.

The `CREATE FAR_SYNC` command

The `CREATE FAR_SYNC` command creates a new far sync instance and adds it to the broker configuration.

```
CREATE FAR_SYNC object-name AS CONNECT IDENTIFIER IS
connect-identifier [DB_RECOVERY_FILE_DEST IS 'directory_location'
DB_RECOVERY_FILE_DEST_SIZE IS 'size' ] [ LOG_FILE_NAME_CONVERT IS
'string-pair-values' ] ;
```

Command Parameter	Description
<i>object-name</i>	The value for <code>DB_UNIQUE_NAME</code> parameter of the far sync instance.
<i>connect-identifier</i>	Connect descriptor or a name to be resolved by an Oracle Net Services naming method.
<i>directory-location</i>	A directory location for the <code>DB_RECOVERY_FILE_DEST</code> initialization parameter of the new far sync instance.
<i>size</i>	A size value for the <code>DB_RECOVERY_FILE_DEST_SIZE</code> parameter of the far sync instance.
<i>string-pair-value</i>	Contains the callout configuration file for the broker configuration named <code>ConfigurationSimpleName</code> . The default name of the callout configuration file is <code>fsfocallout.ora</code> .

Data Guard Broker allows users to create and add a far sync instance to a Data Guard Broker configuration using the `CREATE FAR_SYNC` command.

Command Parameters

object-name: The value for the `DB_UNIQUE_NAME` initialization parameter of the far sync instance.

connect-identifier: A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the initial value of the `DGConnectIdentifier` database property.

directory-location: A directory location value for the `DB_RECOVERY_FILE_DEST` initialization parameter of the new far sync instance. This is an optional parameter.

size: A size value for the `DB_RECOVERY_FILE_DEST_SIZE` initialization parameter of the far sync instance. If this command parameter is specified, the `<directory-location>` command parameter must be also specified. This is an optional command parameter.

string-value-pairs: A list of string pairs. The value is set to the `LOG_FILE_NAME_CONVERT` initialization parameter. In addition, the value is also set to the value of `PARAMETER_VALUE_CONVERT` clause when the `RMAN DUPLICATE` command is invoked. The `PARAMETER_VALUE_CONVERT` clause replaces the first string with the second string in all matching initialization parameter values. Multiple pairs of strings may be specified by this parameter. For example, ' "string1", "string2", "string3", "string4", ...' Where:

- `string1` is the pattern of initialization parameters of the primary database.
- `string2` is the pattern of initialization parameters of the far sync instance.
- `string3` is the pattern of initialization parameters of the primary database.
- `string4` is the pattern of initialization parameters of the far sync instance.

This is an optional argument. If not specified, a copy of the server parameter file on the primary database will be used on far sync instance without any modification. The following graphic shows an example of the `CREATE FAR_SYNC` command.

CREATE FAR_SYNC Command Example

```

DGMGRL> CREATE FAR_SYNC bostonFS
AS CONNECT IDENTIFIER IS "bostonFS_conn_str"
DB_RECOVERY_FILE_DEST IS "$ORACLE_HOME/dbs/"
DB_RECOVERY_FILE_DEST_SIZE IS "100G"
LOG_FILE_NAME_CONVERT ("boston", "bostonFS");
Creating a far sync instance "bostonFS".
...
connected to target database:
boston (DBID=2001456161)
connected to auxiliary database: bostonFS
(not mounted)
RMAN> run {
2> DUPLICATE TARGET DATABASE FOR FARSYNC
3> FROM ACTIVE DATABASE
4> SPFILE
5> PARAMETER_VALUE_CONVERT "boston", "bostonFS"
6> SET LOG_FILE_NAME_CONVERT "boston", "bostonFS"
7> SET DB_UNIQUE_NAME "bostonFS"
8> SET DB_RECOVERY_FILE_DEST_SIZE "100G"
9> SET DB_RECOVERY_FILE_DEST "$ORACLE_HOME/dbs/"
10> SET SGA_TARGET "300MB"
11> RESET SGA_MAX_SIZE
12> RESET DB_CACHE_SIZE
13> RESET LOG_BUFFER
14> RESET SHARED_POOL_SIZE
15> RESET LARGE_POOL_SIZE
16> RESET JAVA_POOL_SIZE
17> RESET STREAMS_POOL_SIZE
18> RESET CONTROL_FILES
19> RESET LOG_ARCHIVE_DEST_1
20> RESET LOG_ARCHIVE_DEST_2
21> RESET CLUSTER_DATABASE
22> NOFILENAMECHECK
23> ;}
Starting Duplicate Db at 08-JAN-19
using target database control file
instead of recovery catalog

```

Data Guard Far Sync instance in Maximum Performance Mode

The far sync instance can fully be utilized in Maximum Performance mode in both normal configurations as well as when fast-start failover (FSFO) is enabled.

This additional flexibility allows RTO/RPO objectives to be met more easily when far sync is utilized in Active Data Guard configurations. Maximum Performance mode provides for a predefined data loss, but with the benefit of a faster automated failover, which is critical in disaster recovery events.

Related Topics

- [Oracle® Data Guard Broker](#)

Fast-Start Failover Callouts

Callout scripts are now available for use with fast-start failover (FSFO). These scripts can contain user-defined commands that are run before and after a FSFO operation.

With the additional flexibility provided by callout scripts, administrators can automate manual actions that must be performed before FSFO events are initiated as well as after a FSFO event has been completed. This provides a more consistent and adaptable configuration reducing the chance for human error to be introduced these events.

Related Topics

- [Oracle® Data Guard Broker](#)

Fast-Start Failover Configuration Validation

Oracle Data Guard Broker now provides early detection of fast-start failover (FSFO) configuration issues and reports those mis-configurations allowing administrators to take action prior to a failover event.

Monitoring and validating a fast-start failover configuration helps maintain and ensure database availability. Potential configuration errors are detected early thereby preventing problems prior to a fast-start failover event that may be required to protect the configuration. This ensures that administrators can confidently maintain and validate their fast-start failover configuration in cases of new deployments as well as updates to an existing configuration.

Related Topics

- [Oracle® Data Guard Broker](#)

PDB Recovery Isolation

PDB Recovery Isolation ensures single PDB recovery while managed standby recovery is recovering other PDBs and prevents administrative PDB operations on the primary database from interfering with recovery of other PDBs on the standby database.

This preserves the PDB isolation principle in regards to Active Data Guard allowing the maintenance, protection, and query SLAs for remaining PDBs to continue unabated which is consistent with how the primary database handles PDB operations.

[Details: PDB Recovery Isolation](#)

This page provide more detailed information about PDB recovery isolation.

Related Topics

- [Oracle® Data Guard Concepts and Administration](#)

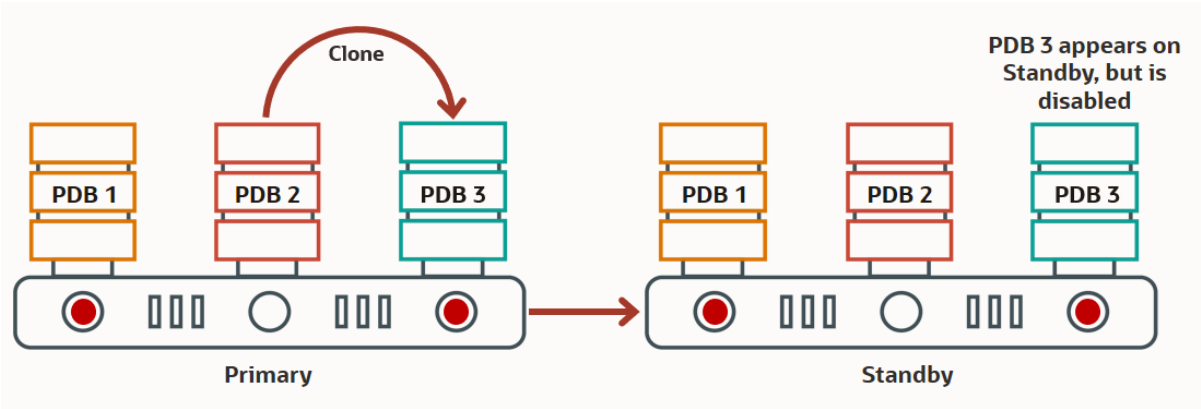
Details: PDB Recovery Isolation

This page provides more detailed information about PDB recovery isolation.

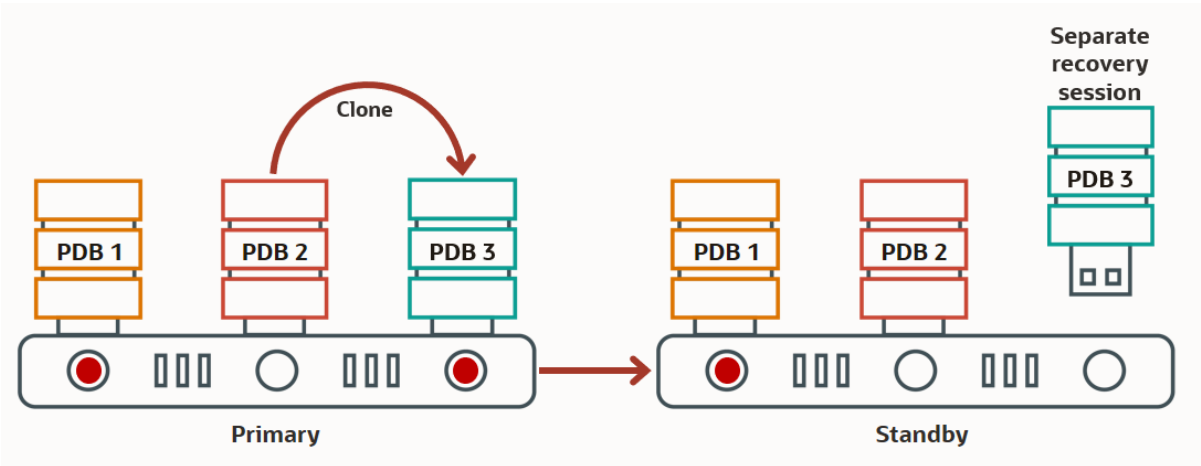
In an Active Data Guard environment, PDB recovery isolation ensures that media recovery of a container database on the standby is not impacted when one or more PDBs are not consistent with the rest of the CDB.

When Active Data Guard media recovery discovers that that a hot cloning, point-in-time recovery, or flashback database operation was performed on a PDB, it performs the following steps:

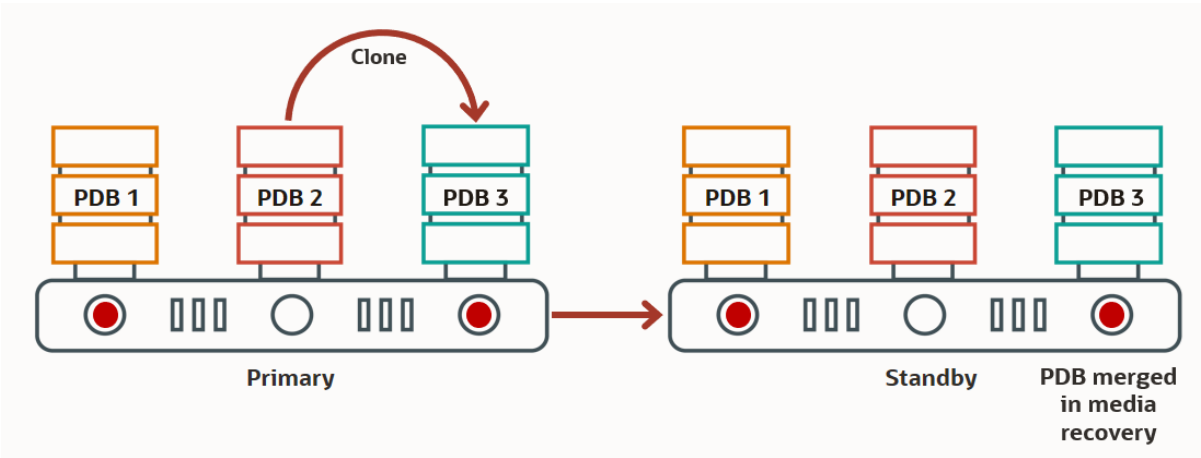
1. Temporarily marks the PDB as disabled and continues with CDB media recovery.



2. Automatically recovers the PDB in a separate background session. This is called PDB recovery isolation.



3. Enables the PDB after the PDB recovery isolation completes and merges the PDB recovery isolation session with the Active Data Guard media recovery session.



Standardized Data Guard Broker Directory Structure

Oracle Data Guard broker now utilizes a standardized directory structure to store client-side files.

Using a standardized directory structure helps keep your Oracle Data Guard environment well organized and consistent. Management is further simplified by using configuration aliases to quickly identify a specific Oracle Data Guard configuration.

Related Topics

- [Oracle® Data Guard Broker](#)

Other features

- [Callout Configuration Scripts](#)
- [Data Guard Broker Managed Default Directory](#)
- [Oracle Database 21c Data Guard Desupported Features](#)
- [The FastStartFailoverLagLimit Configuration Property](#)
- [The PREPARE DATABASE FOR DATA GUARD command](#)
- [The VALIDATE FAST_START FAILOVER Command](#)

Callout Configuration Scripts

This slide explains how callout configuration scripts can be used to execute specified tasks before and after a fast-start failover operation.

Callout Configuration Scripts

Callout configuration scripts can be used to automatically execute specified tasks before and after a fast-start failover operation.

The name of the callout configuration file is `fsfocallout.ora` and is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/callout` directory.

You can create two callout configuration scripts, a *pre-callout* configuration script and *post-callout* configuration script.

To perform specified actions before or after a fast-start failover operation:

1. Create a pre-callout script, or a post-callout script, or both.
2. Create or update the fast-start failover callout configuration file and include the names of the scripts created in the previous step.

Callout configuration scripts can be used to automatically execute specified tasks before and after a fast-start failover operation. The name of the callout configuration file is `fsfocallout.ora`. You cannot use a different name for this file. This file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/callout` directory. If the `DG_ADMIN` environment variable is not defined, or the directory specified by this variable does not exist, or the directory does not have the required permissions, fast-start failover callouts will fail.

The name of the callout configuration scripts is specified in `fsfocallout.ora`. These scripts must be in the same directory as the callout configuration file. You can create two callout configuration scripts, a pre-callout configuration script and post-callout configuration script. Before a fast-start failover operation, the observer

checks if a fast-start failover configuration file exists. If it exists, and it contains a pre-callout script location, this script is run before the fast-start failover is initiated. After fast-start failover succeeds, if a post-callout script is specified in the fast-start failover configuration file this script is run.

The `VALIDATE FAST_START FAILOVER` command parses the callout configuration scripts and checks for errors or misconfigurations. To perform specified actions before or after a fast-start failover operation:

1. Create a pre-callout script, or a post-callout script, or both.
2. Create or update the fast-start failover callout configuration file and include the names of the scripts created in the previous step.

Callout Configuration Script Example

```
# The pre-callout script that is run before fast-start failover is enabled.
FastStartFailoverPreCallout=fsfo_precallout

# The timeout value (in seconds) for pre-callout script
FastStartFailoverPreCalloutTimeout=1200

# The name of the suc file created by the pre-callout script.
FastStartFailoverPreCalloutSucFileName=fsfo_precallout.suc

# The name of the error file that the pre-callout script creates
FastStartFailoverPreCalloutErrorFileName=precallout.err

Action taken by observer if the suc file does not exist after
# FastStartFailoverPreCalloutTimeout seconds or if an error file is
detected before FastStartFailoverPreCalloutTimeout seconds passed
FastStartFailoverActionOnPreCalloutFailure=STOP

# The post-callout script that is run after fast-start failover succeeds
FastStartFailoverPostCallout=fsfo_postcallout
```

The example above displays the contents of the fast-start failover configuration file named `/home1/dataguard/config_NorthSales/callout/fsfocallout.ora`. The `fsfo_preca`llout and `fsfo_postca`llout callout configuration scripts are stored in the same location as `fsfocallout.ora` with the required permissions.

Related Topics

- [Oracle® Data Guard Broker](#)

Data Guard Broker Managed Default Directory

This slide explains how to configure and administer the Data Guard Broker managed default directory.

Data Guard Broker Managed Default Director

Starting with Oracle Database Release 21c, the `DG_ADMIN` environment variable is used to specify the default location for client-side broker files.

Client-side broker files include the:

- Observer configuration file, `observer.ora`
- Observer log file
- Fast-start failover log file, `fsfo.dat`
- Fast-start failover callout configuration file

The files are stored in subdirectories created under `DG_ADMIN`.

You must create the directory specified in `DG_ADMIN` and ensure that it has the required permissions.

Client-side broker files include the observer configuration file (`observer.ora`), observer log file, fast-start failover log file (`fsfo.dat`), and fast-start failover callout configuration file.

Starting with Oracle Database Release 21c, use the `DG_ADMIN` environment variable to specify the default location for client-side broker files. The files are stored in subdirectories created under `DG_ADMIN`. You must create the directory specified in `DG_ADMIN` and ensure that it has the required permissions.

Content of Default Directory for Client-side Files

`admin`: Contains the observer configuration file used by DGMGRL to manage multiple observers. This file also declares broker configurations and defines configuration groups used by multiple configuration commands. The default name of the observer configuration file is `observer.ora`. When DGMGRL starts, if the `DG_ADMIN` environment variable is set and the specified directory has the required permissions, the `admin` directory is created under `DG_ADMIN`. When commands that need access to the observer configuration file are run, such as `START OBSERVING`, `STOP OBSERVING`, `SET MASTEROBSERVER TO`, and `SET MASTEROBSERVERHOSTS`, DGMGRL reports an error if the directory does not have the required permissions.

`config_ConfigurationSimpleName`: Stores files related to the observer and callout configuration. This directory has the same permissions as its parent directory. For each broker configuration on which one or more observers are registered, a directory named `ConfigurationSimpleName` is created. `ConfigurationSimpleName` represents an alias of the broker configuration name. Subdirectories within this directory are used to store the files related to the configuration. This directory is created when you run the `CONNECT` command. When running the `START OBSERVER` command, if this directory does not have the required permissions, DGMGRL reports an error.

`config_ConfigurationSimpleName/log`: Contains the observer log file for the broker configuration named `ConfigurationSimpleName`. The default name of the observer log file is `observer_hostname.log`.

`config_ConfigurationSimpleName/dat`: Contains the observer runtime data file for the broker configuration named `ConfigurationSimpleName`. The default name of the observer runtime data file is `fsfo_hostname.dat`. This file contains important information about the observer. In the event of a crash, data in this file can be used to restart the observer to the status before the crash.

`config_ConfigurationSimpleName/callout`: Contains the callout configuration file, precallout script,

post-callout script, and precallout success file for the broker configuration named `ConfigurationSimpleName`. The default name of the callout configuration file is `fsfocallout.ora`.

Permissions Required by the DG_ADMIN Directory

On Linux/Unix platforms, the directory specified by the `DG_ADMIN` environment variable must have read, write, and execute permissions for the directory owner only. The subdirectories that DGMGRL creates under this directory will also have the same permissions.

On Windows platforms, the directory specified by the `DG_ADMIN` environment variable must have exclusive permissions wherein it can be accessed only by the current operating system user who is running DGMGRL. The subdirectories created under this directory by DGMGRL will also have the same permissions.

If the `DG_ADMIN` environment variable is not set or the specified directory does not exist or the permissions are incorrect, then broker does the following:

- Stores the fast-start failover log file and observer configuration file in the current working directory
- Uses standard output for displaying the observer logs

Every time DGMGRL starts, it checks if the default directories for client-side files exist and if they do not exist, the subdirectories are created.

- If a path name and file name are explicitly specified for client-side files, the specified values are used.
- If only a file name is specified, the file is stored in an appropriate directory under the broker's `DG_ADMIN` directory.
- If only a path is specified, the files are stored in the specified path using the default file names.
- If `DG_ADMIN` is not set, the then files are stored in the current working directory.

If the `DG_ADMIN` environment variable is not set or the specified directory does not exist or the permissions are different from the ones specified above, then broker does the following:

- Stores the fast-start failover log file and observer configuration file in the current working directory
- Uses standard output for displaying the observer logs

Every time DGMGRL starts, it checks if the default directories for client-side files exist. If they do not exist, the subdirectories are created. When you run DGMGRL commands, if a path name and file name are explicitly specified for client-side files, the specified values are used. If only a file name is specified, the file is stored in an appropriate directory under the broker's `DG_ADMIN` directory. If only a path is specified, the files are stored in the specified path using the default file names. If `DG_ADMIN` is not set, the then files are stored in the current working directory.

Related Topics

- [Oracle® Data Guard Broker](#)

Oracle Database 21c Data Guard Desupported Features

This slide lists Data Guard and Data Guard Broker desupported features.

21c Data Guard Broker Desupported Features

The following parameters that were deprecated in Oracle Database Release 19c are now desupported:

```
ArchiveLagTarget LsbyMaxEventsRecorded  
DataGuardSyncLatency LsbyMaxServers  
LogArchiveMaxProcesses LsbyMaxSga  
LogArchiveMinSucceedDest LsbyPreserveCommitOrder  
LogArchiveTrace LsbyRecordAppliedDdl  
StandbyFileManagement LsbyRecordSkipDdl  
DbFileNameConvert LsbyRecordSkipErrors  
LogArchiveFormat LsbyParameters  
LogFileNamesConvert
```

The FastStartFailoverLagLimit Configuration Property

This slide describes the functionality of the new `FastStartFailoverLagLimit` configuration property.

FastStartFailoverLagLimit Configuration Property	
Category	Description
Datatype	Integer
Valid value	Integral number of seconds. Must be greater than, or equal to, 5.
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the primary database after fast-start failover has been enabled.
Cloud control name	Lag Limit

When no synchronous standby destinations are available, a standby that uses asynchronous redo transport can be used as a fast-start failover target provided the `FastStartFailoverLagLimit` configuration property is set. When a synchronous standby becomes available, the broker automatically switches back to the synchronous standby.

The `FastStartFailoverLagLimit` configuration property establishes an acceptable limit, in seconds, that the standby is allowed to fall behind the primary in terms of redo applied. If the limit is reached, then a fast-start failover is not allowed. The lowest possible value is 5 seconds. This property is used when fast-start failover is enabled and the configuration is operating in maximum performance mode.

Related Topics

- [Oracle® Data Guard Broker](#)

The PREPARE DATABASE FOR DATA GUARD command

This slide explains how to configure a database for Data Guard using a single command.

The **PREPARE DATABASE FOR DATA GUARD** command

The **PREPARE DATABASE FOR DATA GUARD** command configures a database for use as a primary database in a Data Guard broker configuration.

```
PREPARE DATABASE FOR DATA GUARD [WITH DB_UNIQUE_NAME IS dbunique-name]
[DB_RECOVERY_FILE_DEST IS directory-location]
[DB_RECOVERY_FILE_DEST_SIZE IS size] [BROKER_CONFIG_FILE_1 IS
brokerconfig-file-1-location] [BROKER_CONFIG_FILE_2 IS
broker-config-file-2-location];
```

You must connect to the primary database as a user with the `SYSDG` or `SYSDBA` privilege.

The **PREPARE DATABASE FOR DATA GUARD** command configures a database and sets it up to be used as a primary database in a Data Guard broker configuration.

Command Parameters

`db_unique_name`: The value for the `DB_UNIQUE_NAME` initialization parameter. If the initialization parameter has been set to a different value, the existing value is replaced with the value specified by `db_unique_name`. If this parameter is not specified, the `DB_UNIQUE_NAME` parameter is set to the value of the `DBNAME` parameter.

`directory-location`: The directory name for the `DB_RECOVERY_FILE_DEST` initialization parameter, which represents the fast recovery area location. The specified directory must be accessible by all instances of a RAC database. This parameter can be omitted if a local archive destination is set. However, if the `DB_RECOVERY_FILE_DEST` initialization parameter has not been set and no local archive destination has been set, specifying this parameter is mandatory. If `directory_location` is specified, a `log_archive_dest_n` initialization parameter is set to the value `USE_DB_RECOVERY_FILE_DEST`. This is done whether or not there is a local archive destination already set.

`size`: A size value for the `DB_RECOVERY_FILE_DEST` initialization parameter. This parameter is mandatory if `DB_RECOVERY_FILE_DEST` is specified.

`broker-config-file-1-location`: A file location that is used to set the `DG_BROKER_CONFIG_FILE1` initialization parameter. The file location specified must be accessible by all instances of a RAC database. This is an optional command parameter.

`broker-config-file-2-location`: A file location that is used to set the `DG_BROKER_CONFIG_FILE2`

initialization parameter. The file location specified must be accessible by all instances of a RAC database. This is an optional command parameter.

PREPARE DATABASE FOR DATA GUARD Command Usage

Database versions starting from Oracle Database 12c Release 2 are supported.

For a single-instance database, if a server parameter file does not exist, it is created using the current in-memory parameter settings and stored in the default location.

This command sets the following initialization parameters, as per the values recommended for the Maximum Availability Architecture (MAA):

```
DB_FILES=1024 DB_FLASHBACK_RETENTION_TARGET=120
LOG_BUFFER=256M PARALLEL_THREADS_PER_CPU=1
DB_BLOCK_CHECKSUM=TYPICAL DG_BROKER_START=TRUE
DB_LOST_WRITE_PROTECT=TYPICAL
```

The **PREPARE DATABASE FOR DATA GUARD** command:

- Enables archivelog mode
- Enables force logging
- Enables Flashback Database
- Sets the RMAN archive log deletion policy to SHIPPED TO ALL STANDBY.

The **PREPARE DATABASE FOR DATA GUARD** command sets the following initialization parameters, as per the values recommended for the Maximum Availability Architecture (MAA):

- `DB_FILES=1024`
- `LOG_BUFFER=256M`
- `DB_BLOCK_CHECKSUM=TYPICAL`

If this value is already set to `FULL`, the value is left unchanged.

- `DB_BLOCK_CHECKSUM=TYPICAL`

If this value is already set to `FULL`, the value is left unchanged.

- `DB_LOST_WRITE_PROTECT=TYPICAL`

If this value is already set to `FULL`, the value is left unchanged.

- `DB_FLASHBACK_RETENTION_TARGET=120`

If this parameter is already set to a non-default value, it is left unchanged.

- `PARALLEL_THREADS_PER_CPU=1`
- `DG_BROKER_START=TRUE`

This command enables archivelog mode, enables force logging, enables Flashback Database, and sets the RMAN archive log deletion policy to SHIPPED TO ALL STANDBY. If standby redo logs do not exist in the primary database, they are added. If the logs exist and are misconfigured, they are deleted and re-created.

Command Example

The following example prepares a database with the name `boston` for use as a primary database. The recovery destination is `$ORACLE_BASE_HOME/dbs`.

```
DGMGRL> PREPARE DATABASE FOR DATA GUARD
```

```
WITH DB_UNIQUE_NAME IS boston
```

```
DB_RECOVERY_FILE_DEST IS "$ORACLE_BASE_HOME/dbs/"
```

```
DB_RECOVERY_FILE_DEST_SIZE is "400G"
```

```
DG_BROKER_CONFIG_FILE1 IS "$ORACLE_HOME/dbs/file1.dat"
```

```
DG_BROKER_CONFIG_FILE2 IS "$ORACLE_HOME/dbs/file2.dat";
```

```
Preparing database "boston" for Data Guard.
```

```
Creating server parameter file (SPFILE) from initialization parameter memory values.
```

```
Database must be restarted after creating the server parameter (SPFILE).
```

```
Shutting down database "boston".
```

```
Database closed.
```

```
Database dismounted.
```

```
ORACLE instance shut down. Starting database "boston" to mounted mode.
```

```
ORACLE instance started.
```

```
Database mounted. Server parameter file (SPFILE) is "ORACLE_BASE_HOME/dbs/spboston.ora".
```

```
Initialization parameter DB_UNIQUE_NAME set to 'boston'.
```

```
Initialization parameter DB_FILES set to 1024.
```

```
Initialization parameter LOG_BUFFER set to 268435456.
```

```
Primary database must be restarted after setting static initialization parameters. Primary database must be restarted to enable archivelog mode.
```

```
Shutting down database "boston".
```

```
Database dismounted.
```

ORACLE instance shut down.

Starting database "boston" to mounted mode.

ORACLE instance started.

Database mounted.

Initialization parameter DB_FLASHBACK_RETENTION_TARGET set to 120.

Initialization parameter DB_BLOCK_CHECKSUM set to 'TYPICAL'.

Initialization parameter DB_LOST_WRITE_PROTECT set to 'TYPICAL'.

Initialization parameter PARALLEL_THREADS_PER_CPU set to 1.

Removing RMAN archivelog deletion policy 1.

Removing RMAN archivelog deletion policy 2.

RMAN configuration archivelog deletion policy set to SHIPPED TO ALL STANDBY.

Initialization parameter DB_RECOVERY_FILE_DEST_SIZE set to '400G'.

Initialization parameter DB_RECOVERY_FILE_DEST set to 'ORACLE_BASE_HOME/ dbs/'. Initialization parameter DG_BROKER_START set to FALSE.

Initialization parameter DG_BROKER_CONFIG_FILE1 set to 'ORACLE_HOME/dbs/ file1.dat'.

Initialization parameter DG_BROKER_CONFIG_FILE2 set to 'ORACLE_HOME/dbs/ file2.dat'.

LOG_ARCHIVE_DEST_n initialization parameter already set for local archival.

Initialization parameter LOG_ARCHIVE_DEST_2 set to 'location=use_db_recovery_file_dest valid_for=(all_logfiles, all_roles)'.

Initialization parameter LOG_ARCHIVE_DEST_STATE_2 set to 'Enable'.

Initialization parameter STANDBY_FILE_MANAGEMENT set to 'MANUAL'.

Standby log group 4 will be dropped because it was not configured correctly.

Standby log group 3 will be dropped because it was not configured correctly.

Adding standby log group size 26214400 and assigning it to thread 1.

Initialization parameter STANDBY_FILE_MANAGEMENT set to 'AUTO'.

Initialization parameter DG_BROKER_START set to TRUE.

Database set to FORCE LOGGING. Database set to ARCHIVELOG.

Database set to FLASHBACK ON.

Database opened.

Related Topics

- [Oracle® Data Guard Broker](#)

The VALIDATE FAST_START FAILOVER Command

This slide explains how to validate a fast-start failover configuration.

The **VALIDATE FAST_START FAILOVER** command enables you to validate a fast-start failover configuration

```
DGMGRL> VALIDATE FAST_START FAILOVER;
Fast-Start Failover: Enabled in Potential Data Loss Mode
Protection Mode: MaxPerformance
Primary: North_Sales
Active Target: South_Sales
Fast-Start Failover Not Possible:
Fast-Start Failover observer not started
Post Fast-Start Failover Issues:
Flashback database disabled for database 'dgv1'
Other issues:
FastStartFailoverThreshold may be too low for RAC databases.
Fast-start failover callout configuration file "fsfocallout.ora"
has the following issues:
Invalid lines
The specified file "./precallout" contains a path.
```

The **VALIDATE FAST_START FAILOVER** command enables you to validate a fast-start failover configuration. It identifies misconfiguration, either while setting up or initiating fast-start failover. This command validates the fast-start failover configuration and reports the following information:

- Incorrectly set up fast-start failover parameters. For example, the fast-start failover threshold is not set appropriately.
- Issues that prevent the enabling or initiating of fast-start failover. This includes issues that prevent the usage of fast-start failover even when the conditions required for fast-start failover are met (for example, fast-start failover is enabled in Observe-Only mode).
- Issues that affect actions taken after fast-start failover is initiated
- Issues that could impact the stability of the broker configuration
- Issues with fast-start failover callout configuration scripts. Displays if the syntax of the fast-start failover configuration file `fsfocallout.ora` is correct and if the pre-callout and post-callout scripts are accessible.

Related Topics

- [Oracle® Data Guard Broker](#)

Flashback

- [Migrate Flashback Data Archive-Enabled Tables Between Different Database Releases](#)
- [Flashback Database Support for Datafile Shrink](#)
- [PDB Point-in-Time Recovery or Flashback to Any Time in the Recent Past](#)

Migrate Flashback Data Archive-Enabled Tables Between Different Database Releases

A new PL/SQL package called `DBMS_FLASHBACK_ARCHIVE_MIGRATE` enables the migration of Flashback Data Archive-enabled tables from a database on any release (in which the package exists) to any database on any release (that supports Flashback Data Archive).

Using the `DBMS_FLASHBACK_ARCHIVE_MIGRATE` PL/SQL package users can export and import the Flashback Archive base tables, along with their history, to another database via the Oracle Transportable Tablespaces capability. Compression is preserved when History Tables enabled with the Advanced Compression Optimization for Flashback Data Archive History Tables capability are migrated.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Flashback Database Support for Datafile Shrink

The existing Flashback Database capability has some limitations with respect to permanent data file resize operations. In earlier releases, the behavior of permanent datafile resizes to a smaller size, (i.e. shrink) on Oracle Databases with Flashback Database enabled, was as follows:

- When a permanent datafile shrink operation is performed on a database, which has Flashback Database enabled, the operation is allowed to succeed. However, any subsequent flashback operations, to a SCN or timestamps across any shrink operations fails (cannot use Flashback Database to undo or rollback a datafile shrink operation).
- When performing a permanent datafile shrink operation on a database, which has Flashback Database enabled and a guaranteed restore point created, the datafile shrink operation fails with a user error.

This new capability is an enhancement to the current Flashback Database feature, allowing Flashback Database operations to succeed with permanent datafile shrinks, and shrinks to succeed even with guaranteed flashback restore points created on the database.

When objects in a tablespace are deleted, or when blocks in objects belonging to the tablespace are defragmented, the tablespace can be shrunk. Shrinking reduces the size of a datafile and returns unused space to the operating system -- including space taken up by UNDO, and defragmenting space in tables, LOBs and etc... The existing Flashback Database capability allowed users to “rewind” the database to a point in the past. However, when a permanent datafile shrink operation was performed users could not use Flashback Database to undo or rollback a datafile shrink operation. This new Flashback Database support for datafile shrink capability enables Flashback Database operations to succeed, with permanent datafile shrinks, and shrinks to succeed even with guaranteed flashback restore points created on the database.

Related Topics

- [Oracle® Database Backup and Recovery User's Guide](#)

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: Flashing Back PDBs to Any Time in the Recent Past](#)

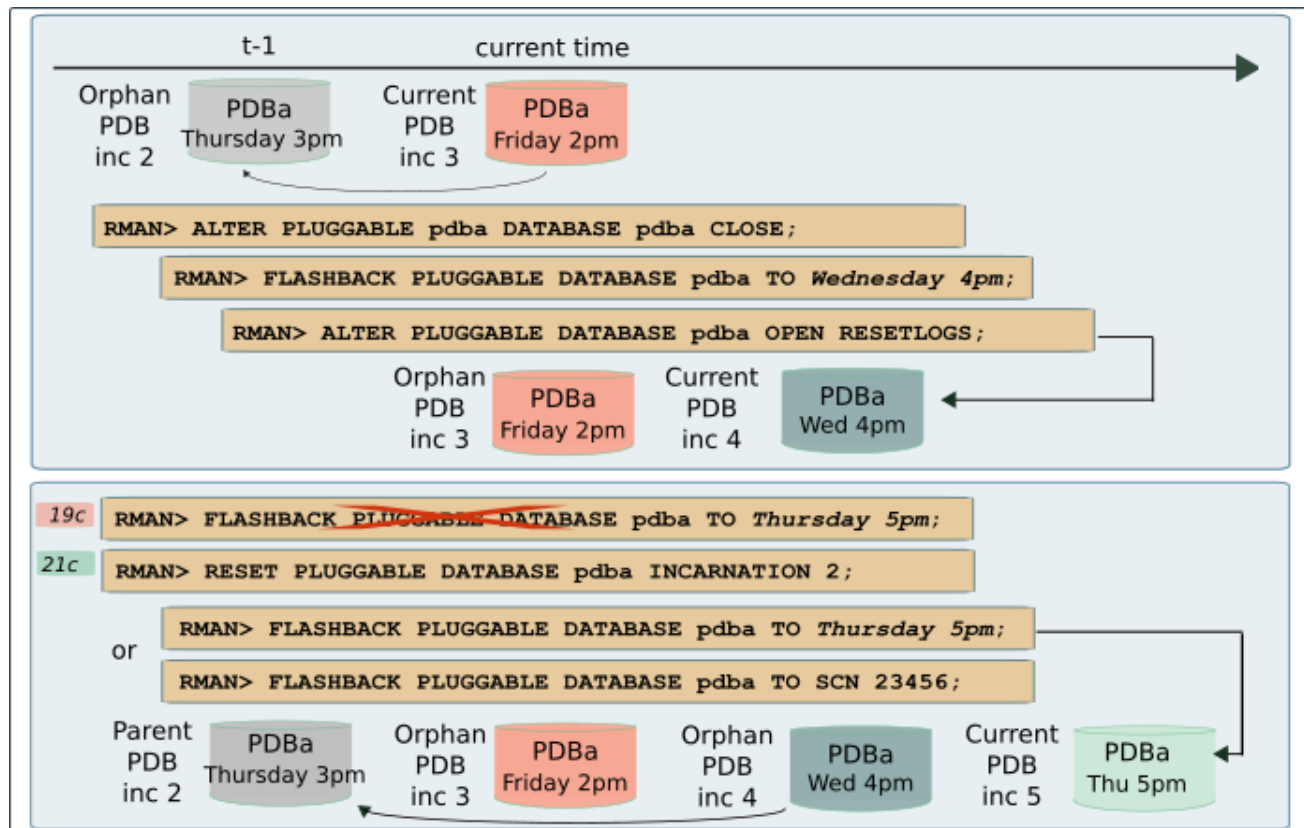
This practice shows how to perform a PDB PITR/Flashback to a specific time, then a PDB PITR/Flashback to a PDB time on an orphan PDB incarnation.

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 PDB PITR or flashback to any time in the recent past.



Oracle database 21c allows point-in-time recovery or flashback of pluggable databases to a point in time which lies on an orphaned PDB branch. This allows you to take the database back 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 an orphan PDB incarnation across multiple DB incarnation: A user can PITR/flash back 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 is that 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: Flashing Back PDBs to Any Time in the Recent Past

Overview

This practice shows how to perform a PDB PITR/Flashback to a specific time and then a PDB PITR/Flashback to a PDB time on an orphan PDB incarnation.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- The shell script enables flashback in the CDB, creates PDB21 and creates the HR schema in PDB21.

```
$ cd /home/oracle/labs/M104782GC10
$ /home/oracle/labs/M104782GC10/setup_Flashback.sh
...
SQL> ALTER DATABASE FLASHBACK on;

Database altered.
...
SQL> exit

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

Connected to:

specify password for HR as parameter 1:

specify default tablespace 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 tbs_for_users
...
Commit complete.

PL/SQL procedure successfully completed.

$
```

- Connect to the CDB root and check that the CDB is open and enabled for flashback.

```

$ sqlplus / AS SYSDBA

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

Connected to:

SQL> SELECT open_mode, flashback_on FROM v$database;

OPEN_MODE          FLASHBACK_ON
-----
READ WRITE        YES

SQL>

```

Step 2 : Generate an error on a table

- Before any DDL or DML command is executed on the HR.EMPLOYEES table in PDB21, display the current SCN, its associated timestamp and the incarnations of the PDB.

```

SQL> CONNECT sys@PDB21 AS SYSDBA
Enter password:
Connected.
SQL> COL TIMESTAMP FORMAT A40
SQL> SELECT CURRENT_SCN, SCN_TO_TIMESTAMP(CURRENT_SCN) "TIMESTAMP" from V$DATA
BASE;

CURRENT_SCN  TIMESTAMP
-----
7139065 16-DEC-20 11.20.27.000000000 AM

SQL> SELECT con_id, status, pdb_incarnation# inc#, begin_resetlogs_scn, end_r
esetlogs_scn
FROM v$pdb_incarnation ORDER BY 3;

CON_ID STATUS          INC# BEGIN_RESETLOGS_SCN END_RESETLOGS_SCN
-----
6 PARENT              0      2621787          2621787
6 CURRENT             0      4691372          4691372

SQL>

```

Possible ORPHAN incarnations would come from previous PDB resetlogs.

- Display the number of rows in the HR.EMPLOYEES table.

```
SQL> SELECT count(*) FROM hr.employees;  
  
COUNT (*)  
-----  
107  
  
SQL>
```

- A user accidentally removes the `HR.EMPLOYEES` table in `PDB21`.

```
SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;  
  
Table dropped.  
  
SQL>
```

Step 3 : Restore the table

- Flash back the PDB to restore the dropped table. Ensure that `PDB21` is closed. Other PDBs can be open and operational.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;  
  
Pluggable database altered.  
  
SQL>
```

- Flash 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 7139065;  
  
Flashback complete.  
  
SQL>
```

- Open `PDB21` with `RESETLOGS`.

```
SQL> ALTER PLUGGABLE DATABASE OPEN RESETLOGS;
```

Pluggable database altered.

```
SQL> SELECT count(*) FROM hr.employees;
```

```

COUNT (*)
-----
          107

```

```
SQL>
```

- Display the incarnations of PDB21.

```
SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn, end_reset
logs_scn
```

```
      FROM v$pdb_incarnation ORDER BY 2;
```

```

CON_ID      INC# STATUS      INCARNATION_SCN  END_RESETLOGS_SCN
-----
          6          0 PARENT          4691372           4691372
          6          4 CURRENT          7139066           7139993

```

```
SQL>
```

Step 4 : Generate a second error on a table

- 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$DATA
BASE;

CURRENT_SCN TIMESTAMP
-----
          7140367 16-DEC-20 11.49.34.000000000 AM

SQL>
```

- Two minutes later, delete 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$DATAFILE;

CURRENT_SCN  TIMESTAMP
-----
7140382 16-DEC-20 11.50.09.000000000 AM

SQL> SELECT con_id, pdb_incarnation# INC#, status, incarnation_scn, end_reset
logs_scn
          FROM v$pdb_incarnation ORDER BY 2;

  CON_ID          INC# STATUS  INCARNATION_SCN  END_RESETLOGS_SCN
-----
         6             0 PARENT          4691372          4691372
         6             4 CURRENT        7139066          7139993

SQL>

```

Step 5 : Restore the table back to the point before the table was dropped

- You decide to flash back the data to the point before the table was dropped.

```

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> FLASHBACK PLUGGABLE DATABASE TO SCN 7139065;

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_reset
logs_scn
      FROM v$pdb_incarnation ORDER BY 2;
  2
  CON_ID          INC# STATUS  INCARNATION_SCN  END_RESETLOGS_SCN
  -----  -----  -----  -----
         6             0 PARENT          4691372           4691372
         6             4 ORPHAN          7139066           7139993
         6             5 CURRENT          7139066           7140876

SQL>

```

- Users ask to reset PDB21 as it was after the salaries were updated and before employee 206 was deleted. This state of pdb21 belongs to incarnation 1 of PDB21. 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 4;
SP2-0734: unknown command beginning "RESET PLUG..." - rest of line ignored.
SQL> EXIT

$

```

This command exists only in RMAN.

```

$ rman TARGET sys@PDB21
target database Password: password
connected to target database: CDB21:PDB21 (DBID=2289122758)

RMAN> LIST INCARNATION OF PLUGGABLE DATABASE pdb21;

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        6        5          2          PDB21     CURRENT  7139066
16-DEC-20          7140876          16-DEC-20
End Reset SCN:7140876          End Reset Time:16-DEC-20          Guid:B693F9369
0D2CB3BE0530200000A6B6F
2        6        4          2          PDB21     ORPHAN   7139066
16-DEC-20          7139993          16-DEC-20
End Reset SCN:7139993          End Reset Time:16-DEC-20          Guid:B693F9369
0D2CB3BE0530200000A6B6F
2        6        0          2          PDB21     PARENT   4691372
10-DEC-20          4691372          10-DEC-20
End Reset SCN:4691372          End Reset Time:10-DEC-20          Guid:B693F9369
0D2CB3BE0530200000A6B6F

RMAN> RESET PLUGGABLE DATABASE pdb21 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 /

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

connected to target database: CDB21 (DBID=2732805675)

RMAN> ALTER PLUGGABLE DATABASE pdb21 CLOSE;

using target database control file instead of recovery catalog
Statement processed

RMAN> RESET PLUGGABLE DATABASE pdb21 TO INCARNATION 4;

```

```

using target database control file instead of recovery catalog
pluggable database reset to incarnation 4

RMAN> FLASHBACK PLUGGABLE DATABASE pdb21 TO SCN 7139066;
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 12/16/2020 11:55:08
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 mi
ddle 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 show
n
// 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 middl
e
// of a previous PDB RESETLOGS operation. If flashback to a SCN on t
he
// 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 when the user increased the salary of the employees.

```
$ rman TARGET /
Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

connected to target database: CDB21 (DBID=2732805675)

RMAN> RESET PLUGGABLE DATABASE pdb21 TO INCARNATION 4;

using target database control file instead of recovery catalog
pluggable database reset to incarnation 4

RMAN> FLASHBACK PLUGGABLE DATABASE pdb21 TO SCN 7140367;
Starting flashback at 16-DEC-20
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=434 device type=DISK

starting media recovery
media recovery complete, elapsed time: 00:00:07

Finished flashback at 16-DEC-20

RMAN> EXIT

Recovery Manager complete.

$
```

- Open the PDB and verify that the data is restored with the employees' salaries updated and employee 206.

```

$ sqlplus sys@PDB21 AS SYSDBA
Enter password: password

Connected to:

SQL> ALTER PLUGGABLE DATABASE pdb21 OPEN RESETLOGS;

Pluggable database altered.

SQL> CONNECT system@PDB21
Enter 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_reset
logs_scn
          FROM v$pdb_incarnation ORDER BY 2;

  CON_ID          INC# STATUS  INCARNATION_SCN  END_RESETLOGS_SCN
-----
         6             0 PARENT          4691372          4691372
         6             4 PARENT          7139066          7139993
         6             5 ORPHAN          7139066          7140876
         6             6 CURRENT          7140368          7142535

SQL> EXIT
$

```

GoldenGate

- [Automatic CDR Enhancements](#)
- [Improved Support for Table Replication for Oracle GoldenGate](#)
- [LogMiner Views Added to Assist Replication](#)
- [Oracle GoldenGate for Oracle and XStream Support for JSON Data Type](#)

Automatic CDR Enhancements

Automatic Conflict Detection and Resolution (CDR) was introduced in Oracle Database 12c Release 2 (and Oracle GoldenGate 12.3.0.1) to automate the conflict detection and resolution configuration in active-active Oracle GoldenGate replication setups. In Oracle Database 21c, we are enhancing automatic CDR to support earliest timestamp based resolution and site priority based resolution.

Active-active Oracle GoldenGate replication customers can use the automatic CDR feature on more types of tables simplifying their active-active replication setups.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Improved Support for Table Replication for Oracle GoldenGate

In earlier releases, extracting tables to Oracle GoldenGate required supplemental logging data to replicate table/schema, and required you to set `TABLE/TABLEEXCLUDE` parameters to configure which table to extract.

Starting with this release, to coordinate with the Oracle GoldenGate feature `OGG EXTRACT`, the `LOGICAL_REPLICATION` clause now provides support for automatic extract of tables.

In addition, two new views, `DBA_OGG_AUTO_CAPTURED_TABLES` and `USER_OGG_AUTO_CAPTURED_TABLES`, provide you with tools to query which tables are enabled for Oracle GoldenGate automatic capture.

Related Topics

- [Oracle® Database Utilities](#)

LogMiner Views Added to Assist Replication

The `DBMS_ROLLING` package contains a new parameter that enables you to block the replication of operations unsupported by Transient Logical Standby.

Starting with this release, in the `DBMS_ROLLING.set_parameter()` procedure, there is a new parameter called `BLOCK_UNSUPPORTED`. By default, `BLOCK_UNSUPPORTED` is set to 1 [YES], indicating that operations performed on tables that are unsupported by Transient Logical Standby will be blocked on the primary database. If set to 0 [OFF], then the `DBMS_ROLLING` package does not block operations on unsupported tables. Those tables will not be maintained by Transient Logical Standby, and will diverge from the primary database.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Oracle GoldenGate for Oracle and XStream Support for JSON Data Type

Oracle GoldenGate for Oracle and XStream supports JavaScript Object Notation (JSON) data type.

JSON data type represents JSON in a proprietary binary format that is optimized for query and DML processing and can yield performance improvements for JSON processing in the database. It provides strong typing of JSON values so that the data type can be propagated through SQL expressions and view columns.

Related Topics

- [Oracle® Database XStream Guide](#)

Multitenant

- [DRCP Enhancements for Oracle Multitenant](#)
- [Expanded Syntax for PDB Application Synchronization](#)
- [MAX_IDLE_BLOCKER_TIME Parameter](#)
- [Namespace Integration with Database](#)
- [Support Per-PDB Capture for Oracle Autonomous Database](#)
- [Time Zone support for PDBs in DBCA](#)
- [Details: Using non-CDBs and CDBs](#)

DRCP Enhancements for Oracle Multitenant

Starting with Oracle Database 21c, you can configure Database Resident Connection Pooling (DRCP) from the CDB to individual PDBs for improved PDB tenancy management.

In previous releases, the DRCP pool was used by the entire container database (CDB). With per-PDB pools, you can now configure, manage, and monitor pools on individual pluggable databases (PDBs), on the basis of tenancy. This feature also provides the ability to specify Connection Class and Purity support in connect strings for DRCP. With this change, you can leverage DRCP without application code change. This feature eases the management of DRCP pool by changing the granularity of the DRCP pool from the entire CDB to a per-PDB DRCP pool. This change enables tenant administrators to configure and manage independent tenant-specific DRCP pools. Additionally, this feature enables applications to leverage some of the DRCP benefits with making any application changes.

Related Topics

- [Oracle® Database Development Guide](#)

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](#)

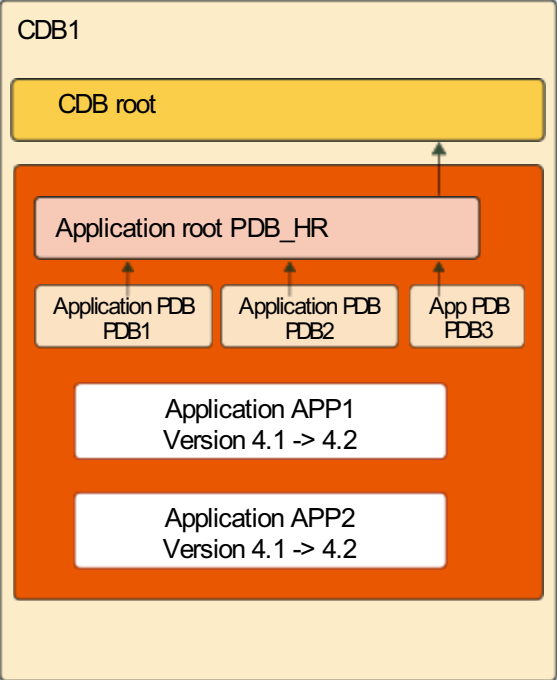
This practice shows how to reduce the number of synchronization statements when you have to synchronize multiple applications in application PDBs. In previous Oracle Database versions, you had to execute as many synchronization statements as applications.

Related Topics

- [Oracle® Multitenant Administrator's Guide](#)

Details: Expanded Syntax for PDB Application Synchronization

This page provides more detailed information about enhancements to applications synchronization in application PDBs.



1- Start the APP1 application upgrade.

```
SQL> CONNECT sys@PDB_HR
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
BEGIN UPGRADE '4.1' TO '4.2';
```

2- Complete the APP1 application upgrade.

```
SQL> @scripts
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
END UPGRADE '4.1' TO '4.2';
```

3- Start the APP2 application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION app2
BEGIN UPGRADE '4.1' TO '4.2';
```

4- Complete the APP2 application upgrade.

```
SQL> @scripts_2
SQL> ALTER PLUGGABLE DATABASE APPLICATION app2
END UPGRADE '4.1' TO '4.2';
```

5- Synchronize both APP1 and APP2 applications.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION
app1, app2 SYNC;
```

In Oracle Database 19c, the `ALTER PLUGGABLE DATABASE APPLICATION ... SYNC` statement accepts only one application name to synchronize with the application root. You have to execute the statement as many times as the number of applications to synchronize with the application root.

In Oracle Database 21c, the `ALTER PLUGGABLE DATABASE APPLICATION ... SYNC` statement 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 `ALTER PLUGGABLE DATABASE APPLICATION apexapp, ordsapp SYNC` statement 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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment

- 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 creates the two application PDBs in the application container.
 - To be able to connect during the shell script execution to `TOYS_ROOT`, `ROBOTS` and `DOLLS`, create entries in the `tnsnames.ora` file as explained in the [Practices Environment](#) recommendations.
 - Execute the shell script.

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

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

Connected to:

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> CREATE PLUGGABLE DATABASE toys_root AS APPLICATION CONTAINER
  2          FROM pcb21 KEYSTORE IDENTIFIED BY password;

Pluggable database created.

SQL> alter PLUGGABLE DATABASE toys_root open;

Pluggable database altered.

SQL> exit

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



```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app begin install '1.0';

Pluggable database altered.

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

SQL> CREATE TABLESPACE toys_tbs DATAFILE SIZE 100M autoextend on next 10
M 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, categor
y 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
```

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

```
Connected to:
```

```
SQL> create pluggable database robots ADMIN USER admin identified by password ROLES=(CONNECT) KEYSTORE IDENTIFIED BY password;
```

```
Pluggable database created.
```

```
SQL> create pluggable database dolls ADMIN USER admin identified by password ROLES=(CONNECT) KEYSTORE IDENTIFIED BY password;
```

```
Pluggable database created.
```

```
SQL>
```

```
SQL> alter pluggable database robots open;
```

```
Pluggable database altered.
```

```
SQL> alter pluggable database dolls open;
```

```
Pluggable database altered.
```

```
SQL>
```

```
SQL> host rm -r /opt/oracle/dcs/commonstore/wallets/tde/$ORACLE_UNQNAME/bak_cwallet
```

```
SQL> host mkdir /opt/oracle/dcs/commonstore/wallets/tde/$ORACLE_UNQNAME/bak_cwallet
```

```
SQL> host mv /opt/oracle/dcs/commonstore/wallets/tde/$ORACLE_UNQNAME/cwallet.sso /opt/oracle/dcs/commonstore/wallets/tde/$ORACLE_UNQNAME/bak_cwallet/
```

```
SQL> conn / as sysdba
```

```
Connected.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE close;
```

```
keystore altered.
```

```
-  
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open IDENTIFIED BY password;  
keystore altered.  
  
SQL>  
SQL> ALTER SESSION SET CONTAINER=TOYS_ROOT;  
  
Session altered.  
  
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open IDENTIFIED BY password;  
keystore altered.  
  
SQL> ALTER SESSION SET CONTAINER=ROBOTS;  
  
Session altered.  
  
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open IDENTIFIED BY password;  
keystore altered.  
  
SQL> administer key management set key identified by password with backu  
p;  
keystore altered.  
  
SQL> ALTER SESSION SET CONTAINER=DOLLS;  
  
Session altered.  
  
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open IDENTIFIED BY password;  
keystore altered.  
  
SQL> administer key management set key identified by password with backu  
p;  
keystore altered.  
  
SQL> alter session set container=CDB$ROOT;  
  
Session altered.  
  
SQL> administer key management create AUTO_LOGIN keystore from keystore  
'/opt/oracle/dcs/commonstore/wallets/tde/$ORACLE_UNQNAME' identified by  
password;  
keystore altered.  
  
SQL> exit
```

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

```
Connected to:
```

```
SQL> shutdown immediate  
Database closed.  
Database dismounted.  
ORACLE instance shut down.  
SQL> exit
```

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

```
Connected to an idle instance.
```

```
SQL> STARTUP  
ORACLE instance started.
```

```
Total System Global Area 851440288 bytes  
Fixed Size 9691808 bytes  
Variable Size 599785472 bytes  
Database Buffers 104857600 bytes  
Redo Buffers 19664896 bytes  
In-Memory Area 117440512 bytes
```

```
Database mounted.
```

```
Database opened.
```

```
SQL> ALTER PLUGGABLE DATABASE all OPEN;
```

```
Pluggable database altered.
```

```
SQL> exit  
$
```

Step 2 : Display the installed applications

- Display the applications that have been installed.

```

$ sqlplus / AS SYSDBA

Connected to:

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 `toys_app` and `sales_toys_app` applications are installed in the application container at version 1.0.

Step 3 : Synchronize the application PDBs

- Synchronize the application PDBs with the new `toys_app` and `sales_toys_app` applications.

```

SQL> CONNECT sys@robots AS SYSDBA
Enter password:
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app, sales_toys_app SYNC;

Pluggable database altered.

SQL>

```

- Display the applications installed in the application container.

```

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

```

SALES_TOYS_APP    1.0          NORMAL    ROBOTS
TOYS_APP          1.0          NORMAL    ROBOTS

```

```
SQL> CONNECT sys@dolls AS SYSDBA
```

```
Enter 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
```

```
$
```

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 the 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`, which is used to influence sessions behavior.

The screenshot displays two terminal windows for Oracle Database 19c. The left window shows a DBA user setting `MAX_IDLE_TIME = 2` and then a user `U1` attempting a `SELECT` query that fails with an error: `ORA-03113: end-of-file on communication channel`. The right window shows a DBA user setting `MAX_IDLE_BLOCKER_LIMIT = 600`, followed by an `UPDATE` statement (107 rows updated), a `DELETE FROM employees` statement, and another `SELECT count(*)` query that fails with the same error. Finally, the right window shows the DBA user setting `MAX_IDLE_BLOCKER_TIME = 600` in Oracle Database 21c.

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. The 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  => 60)
```

- The `MAX_IDLE_TIME` initialization parameter 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. The default is NULL, which implies unlimited.

Oracle Database 21c allows you to set the `MAX_IDLE_BLOCKER_TIME` initialization parameter 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 the MAX_IDLE_BLOCKER_TIME Parameter

Overview

This practice shows how to terminate a blocking session by using the new `MAX_IDLE_BLOCKER_TIME` initialization parameter.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set up the environment with two sessions

- Prepare two terminal sessions, one logged in to PDB21 as HR and another one logged in to PDB21 as SYST

```
$ sqlplus system@PDB21
Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter password:

Connected to:

SQL> SET SQLPROMPT "SQL system> "
SQL system>
```

- Log in to PDB21 as HR.

```
$ sqlplus hr@PDB21
Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter password:

Connected to:

SQL> SET SQLPROMPT "SQL hr> "
SQL hr>
SQL>
```

Step 2: Set `MAX_IDLE_BLOCKER_TIME` to two minutes

- In the SYSTEM session, set the `MAX_IDLE_BLOCKER_TIME` initialization parameter 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>

```

Step 3 : Test

- In the HR session, update the employees' salary.

```

SQL hr> UPDATE hr.employees SET salary=salary*2;

107 rows updated.

SQL hr>

```

- 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> COMMIT;

Commit completed.

SQL system>

```

- 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 a resource for longer than two minutes.

Step 4 : Examine the DIAG trace files

- View the DIAG trace file.

```
$ cd /u01/app/oracle/diag/rdbms/cdb21*/CDB21/trace
$ grep -i 'Session with ID' *
CDB21_dia0_17197_base_1.trc:HM: Session with ID 197 serial # 34111 (FG) on si
ngle instance 1 is hung
CDB21_dia0_17197_base_1.trc:HM: Session with ID 426 serial # 29909 (FG) on si
ngle instance 1 is hung
CDB21_dia0_17197_base_1.trc:HM: Session with ID 197 with serial number 34111
is no longer hung
CDB21_dia0_17197_base_1.trc:HM: Session with ID 426 with serial number 29909
is no longer hung
$ cat CDB21_dia0_17197_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
-----
Hang Hang          Root      Chain Total      Hang      Hang      Hang
ID  Type Status    Inst Root #hung #hung  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

Total Self-          Total Total  Outlr  Outlr  IO
Hung  Rslvd  Rslvd  Wait WaitTm  Wait WaitTm  Outlr
Sess  Hangs  Hangs  Count Secs  Count Secs  Count Wait Event
-----
   1    0    0    0    0    0    0    0    0 enq: TX - row lock co
ntention
...
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

$
$

```

You can also view the PMON trace file.

```

$ grep -i 'Kill idle blocker' *
CDB21_pmon_17088.trc:Kill idle blocker, hang detected
$ cat cat CDB21_pmon_17088.trc

*** 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 id
le=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
...
$

```

- In the SYSTEM session, query the employees' commission percentage.

```

SQL system> SELECT DISTINCT commission_pct FROM hr.employees;

COMMISSION_PCT
-----
              0

SQL system> EXIT
$

```

Namespace Integration with Database

Database Nest is an infrastructure that provides operating system resource isolation and management, file system isolation, and secure computing for CDBs and PDBs. This infrastructure enables a database instance to run in a protected, virtualized environment.

Sharing instance-level and operating system resources can lead to security and isolation constraints, especially in large-scale cloud deployments. Vulnerabilities can be external, such as compromised applications, unauthorized access of resources, and shared resources. An example of an internal vulnerability is a compromised Oracle process.

Database Nest isolates a database instance from other databases and applications running on the same host, and also isolates PDBs from each other and from the CDB. The feature is implemented as a Linux-specific package that provides hierarchical containers, called nests. A CDB resides within a single parent nest, while PDBs reside within the individual child nests created within the parent.

Linux processes in a PDB nest have their own process ID (PID) number spaces and cannot access PIDs in other nests. Process isolation provides a last level of defense in a security breach if a malicious user compromises a process.

[Details: Database Nest](#)

This page provide more detailed information about Database Nest, also known as DbNest.

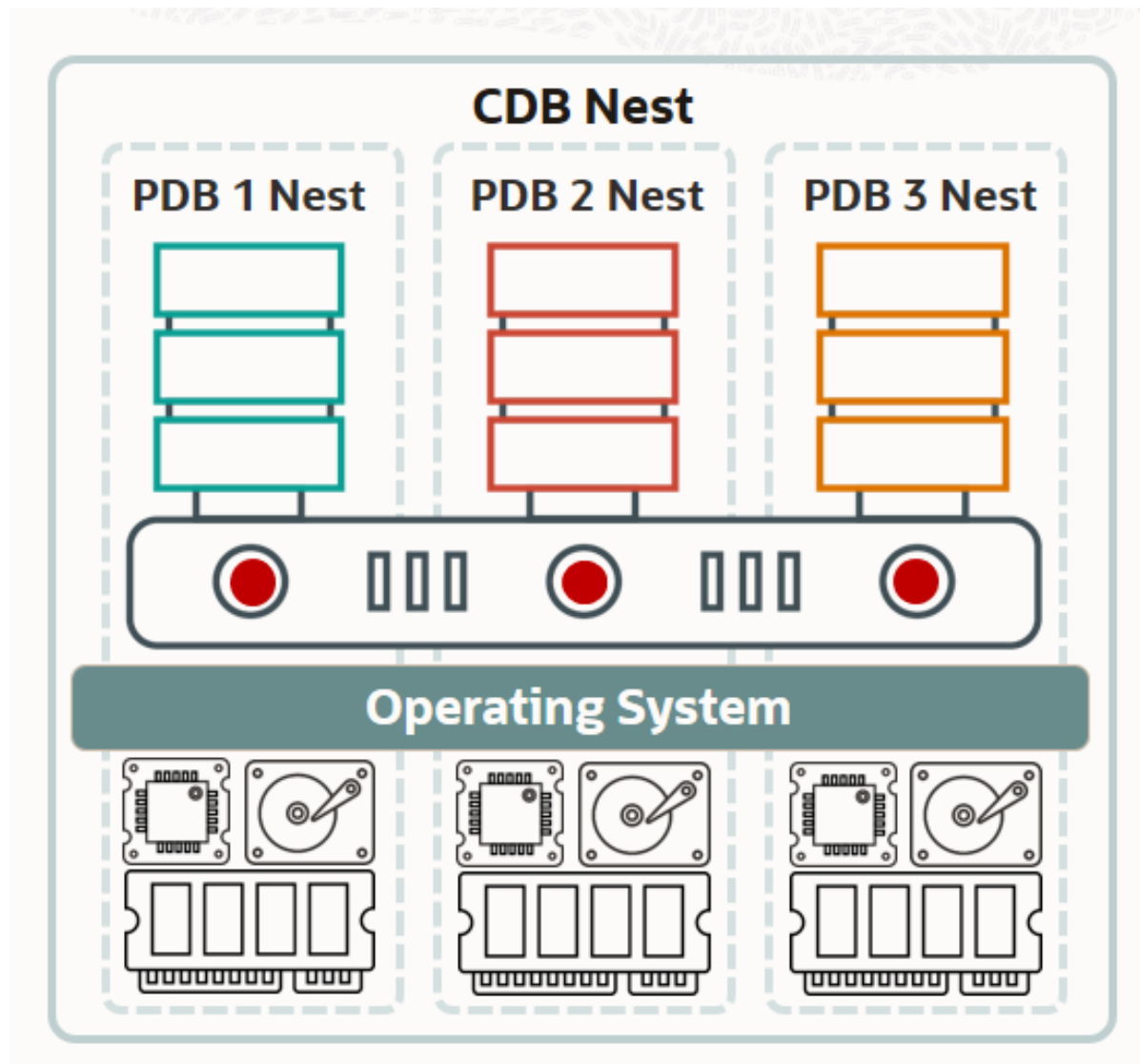
Related Topics

- [Oracle® Database Security Guide](#)

Details: Database Nest

This page provides more detailed information about Database Nest, also known as DbNest.

Database Nest provides hierarchical, isolated run-time environments at the CDB and PDB level. Database Nest protects a multitenant environment from security breaches by using the latest Linux resource isolation, namespace, and control group features.



A nest is a runtime environment that Oracle Database creates for every CDB, PDB, or application container. Each nest corresponds to exactly one container. The nest hierarchy exactly mirrors the container hierarchy. Because a CDB can contain one or more PDBs, a parent CDB nest can have one or more child nests. Each child nest corresponds to the PDB that can be contained in the nest.

Support Per-PDB Capture for Oracle Autonomous Database

To securely capture and replicate individual pluggable database (PDB) changes to Oracle Autonomous Database, you can now use Oracle GoldenGate to provide per-PDB capture.

You can now provide local user credentials to connect to an individual PDB in a multitenant architecture Oracle Database, and replicate the data from just that PDB to an Oracle Autonomous Database. You no longer need to create a common user with access to all PDBs on the multitenant container database (CDB) to replicate a PDB to an Oracle Autonomous Database. Instead, you can now provision a local user with a predefined set of privileges to the source PDB that you want to capture. All LogMiner and Capture processing takes place only in this PDB, and only data from this specific PDB is captured and written to the Oracle GoldenGate trail. As part of this feature, the behavior for V\$LOGMNR_CONTENTS changes, depending on whether you connect to a PDB, or connect to the CDB\$ROOT.

Related Topics

- [Oracle® Database Utilities](#)

Time Zone support for PDBs in DBCA

In Database Configuration Assistant (DBCA) silent mode, you can optionally use the `-pdbTimezone` parameter with the `-createPluggableDatabase` and `-configurePluggableDatabase` commands to specify a time zone for a pluggable database (PDB).

Related Topics

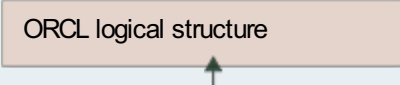
- [Oracle® Multitenant Administrator's Guide](#)

Details: Using non-CDBs and CDBs

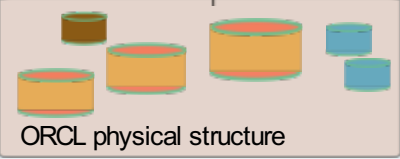
This page provides information about the availability of CDBs only in Oracle Database 21c. The non-CDB architecture was deprecated in Oracle Database 12c. It is desupported in Oracle Database 21c which means that the Oracle Universal Installer and DBCA can no longer be used to create non-CDB Oracle Database instances.

21c Creating non-CDBs **not supported any longer**

ORCL logical structure



ORCL physical structure

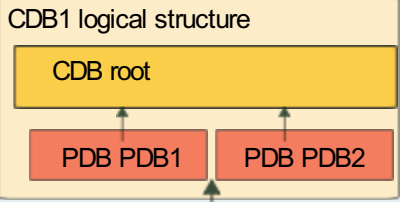


```
$ dbca -silent -createDatabase -templateName General_Purpose.dbc
-gdbname ORCL -sid ORCL
-createAsContainerDatabase false ...

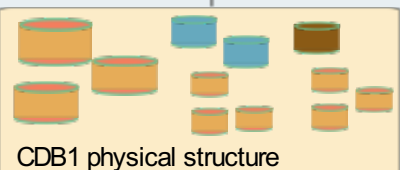
[FATAL] [DBT-10333] Container database (CDB) creation option
is not selected.
CAUSE: Non-CDB creation is not supported.
ACTION: Make sure container database (CDB) option is selected.
$
```

Creating CDBs only

CDB1 logical structure



CDB1 physical structure



```
$ dbca -silent -createDatabase -templateName General_Purpose.dbc
-gdbname ORCL -sid ORCL -totalMemory 1800
-sysPassword "password" ...
```

- Not necessary to use `-createAsContainerDatabase` clause any longer
- No PDB created by default

A multitenant container database is the only supported architecture in Oracle Database 21c.

Oracle Real Application Clusters (RAC)

- [Cache Fusion Hardening](#)
- [Database Management Policy change for Oracle RAC in DBCA](#)
- [Integration of PDB as a Resource in Clusterware](#)
- [Pluggable Database Cluster Resources](#)

Cache Fusion Hardening

The Global Cache Service (LMS) process is vital to the operation of an Oracle Real Application Clusters (Oracle RAC) Database. Cache Fusion Hardening helps to ensure that the critical LMS process remains running despite some discrepancies between instances that would otherwise lead to a LMS and consequently database instance failures.

Cache Fusion Hardening increases availability by reducing outages, particularly in consolidated environments in which multiple Pluggable Databases (PDBs) are operated in the same Oracle RAC Container Database.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Details: Cache Fusion Hardening

This page provides more information about the new cache fusion hardening features.

21c Cache Fusion Hardening

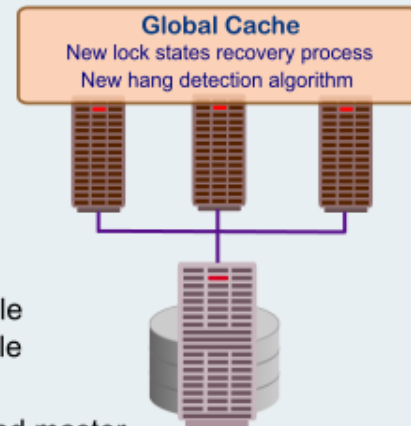
In Real Application Clusters, when a fatal error affects a critical background process such as LMS, the instance is terminated.

When this happens, the workload may be delayed when redirected to surviving instances while waiting for instance reconfiguration and recovery to be completed.

In cloud environment with multiple PDB running in a single RAC instance, a single instance outage can affect multiple PDBs.

Most errors are caused by discrepancies among client and master instances and can be reconciled through a new special lock states recovery process.

A process monitoring long pending requests detects cache fusion hangs and initiates steps to determine inconsistencies among instances and resolve the issue.



In RAC, for each cache fusion lock, there is a set of client instances and a single master instance. LMS is the main process at both client and master instances handling cache fusion requests. Most of the cache fusion protocol traffic will pass through LMS processes. There is significant communication between the client and master instances and some communication through remote block transfer between any two instances. All the communication carries some lock state information and the information will eventually carry back to master or client instances. The majority of errors encountered by LMS are caused by some discrepancies among the instances such as missing or stale last disk write SCN and missing or delayed inter-instance messages among instances. This feature introduces a way to determine differences and to recover from such inconsistencies, preventing crashing the critical process LMS which will result in an instance termination. Similar lock state reconciliation can be applied to other fatal processes involved in cache fusion protocol.

Some of the discrepancies in lock states among instances can sometimes cause cache fusion requests to hang. For example, for cache fusion write requests, the master may have thought it granted a write permission when receiving a write request from the same instance which the master has picked as the writer. The master may ignore the write request as it may have crossed paths with a write permission message sent by the master due to a write request from another instance. If for some reason, the first write permission sent by the master is not received by the writer instance, it may cause a cache fusion write hang which may result in a checkpointing hang. A mechanism monitors long pending requests to detect this kind of cache fusion hang and the protocols to address the inconsistencies among instances are triggered to resolve the issue.

Database Management Policy change for Oracle RAC in DBCA

DBCA supports creation of database management policy for Oracle RAC.

The following variants are supported: `Automatic` and `RANK`.

The feature allows customers to modify the PDB placement in an Oracle RAC database.

Related Topics

- [Oracle® Multitenant Administrator's Guide](#)

Integration of PDB as a Resource in Clusterware

Integration of pluggable databases (PDBs) as a resource in Oracle Clusterware completes the integration of PDBs as Oracle Clusterware resources. This feature includes support using utilities and command line tools.

With resources to be defined and mapped at the PDB level, rather than the previously supported CDB level, you are better able to manage PDBs, the workload and resources against those PDBs, and monitor those PDBs.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

Pluggable Database Cluster Resources

Pluggable Database (PDB) Cluster Resources enables direct mapping and control of the PDB resources. Unlike in previous versions, in which cluster resources for Multitenant databases were mapped against the Container Database (CDB) using a control of Pluggable Databases (PDBs) using services.

Pluggable Database (PDB) Cluster Resources enable a tighter and more effective control of PDBs in an Oracle RAC Database.

Related Topics

- [Oracle® Real Application Clusters Administration and Deployment Guide](#)

SecureFiles

- [SecureFiles Shrink](#)

SecureFiles Shrink

SecureFiles Shrink provides a way to free the unused space in SecureFiles segments while allowing concurrent reads and writes to SecureFiles data.

SecureFiles Shrink also helps to reduce fragmentation and thereby improve the read and write performance. The feature supports all types of SecureFiles LOBs - compressed, deduplicated and encrypted.

[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 defragmentation operations on SecureFiles.

19c SecureFiles fragmentation impacts:

- Sequential read performance of the LOBs
- Efficiency of space layer to search for free space and the overall performance of SecureFile DMLs

`SQL> ALTER TABLE tab_containing_LOBs SHRINK SPACE CASCADE;` Not supported

`SQL> ALTER TABLE tab_containing_LOBs MOVE LOB ...;` Expensive operation in time and space

21c SecureFiles defragmentation shrinks the space used by SecureFiles segments without compromising concurrent access to SecureFiles data.

- Partitioned, including compression, encryption, and deduplication
- Allows concurrent queries and DMLs, and serializes concurrent DDLs
- Works on RAC
- Restarts and recovers from failure or interruption
- Honors LOB retention

`V$SECUREFILE_SHRINK`

`SQL> ALTER TABLE tab1 MODIFY LOB (lob_column1) (SHRINK SPACE);`

`SQL> ALTER TABLE tab_containing_LOBs SHRINK SPACE CASCADE;`

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

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create a table with a SecureFile LOB

- Execute a shell script that creates a tablespace with sufficient space to let the LOB grow and be a candidate for shrinking.

```

$ cd /home/oracle/labs/M104780GC10
$ /home/oracle/labs/M104780GC10/setup_LOB.sh
...
SQL> DROP TABLESPACE tbs_for_users INCLUDING CONTENTS AND DATAFILES CASCADE C
ONSTRAINTS;

Tablespace dropped.

SQL> CREATE TABLESPACE tbs_for_users DATAFILE SIZE 500M;

Tablespace created.

SQL> create user hr identified by password default tablespace tbs_for_users;

User created.

SQL> grant dba to hr;

Grant succeeded.

SQL> exit

$

```

- Create a table with a CLOB column in PDB21.

```

$ sqlplus system@PDB21

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

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

Connected to:

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

Table created.

SQL>

```

Step 2 : Shrink the SecureFile LOB after rows are inserted and updated

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


```

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
-----
          76063 COMPLETE
10-NOV-20 11.30.55.545 AM +00:00
10-NOV-20 11.30.55.917 AM +00:00
           2           2           2           1           1
           1           3
SQL>

```

As a result, two blocks are freed.

Step 3 : Shrink the SecureFile LOB after rows are updated

- 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>
```

- Shrink the LOB segment.

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

SQL>
```

- Display the number of extents or blocks freed.


```

SQL> SELECT * FROM v$securefile_shrink WHERE LOB_OBJD=76063;

  LOB_OBJD SHRINK_STATUS
-----
START_TIME
-----
END_TIME
-----
BLOCKS_MOVED BLOCKS_FREED BLOCKS_ALLOCATED EXTENTS_ALLOCATED EXTENTS_FREED
-----
EXTENTS_SEALED      CON_ID
-----
      76063 COMPLETE
10-NOV-20 11.32.57.963 AM +00:00
10-NOV-20 11.33.01.828 AM +00:00
      2648      2648      2648      1      11
      11      3

      76063 COMPLETE
10-NOV-20 11.30.55.545 AM +00:00
10-NOV-20 11.30.55.917 AM +00:00
      2      2      2      1      1
      1      3

SQL>

```

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

- Update the CLOB.

```

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

8 rows updated.

SQL> COMMIT;

Commit complete.

SQL>

```

- Shrink the LOB segment.

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

Table altered.

SQL>
```

- Display the number of extents or blocks freed.

```
SQL> SELECT * FROM v$securefile_shrink WHERE LOB_OBJD=76063;

  LOB_OBJD SHRINK_STATUS
-----
START_TIME
-----
END_TIME
-----
BLOCKS_MOVED BLOCKS_FREED BLOCKS_ALLOCATED EXTENTS_ALLOCATED EXTENTS_FREED
-----
EXTENTS_SEALED      CON_ID
-----
      76063 COMPLETE
15-DEC-20 01.08.44.404 PM +00:00
15-DEC-20 01.08.48.270 PM +00:00
          2648          2648          2648          1          11
           11           3
      76063 COMPLETE
15-DEC-20 01.09.22.785 PM +00:00
15-DEC-20 01.09.30.739 PM +00:00
          5523          5523          5523          1          19
           19           3

SQL> EXIT
$
```

As a result, 5523 blocks are freed. Observe that only the row of the previous shrinking operation is kept.

Sharding

- [Centralized Backup and Restore of a Sharded Database](#)
- [Create a Sharded Database from Multiple Existing Databases \(Federated Sharding\)](#)
- [Multi-Shard Query, Data Loading, and DML Enhancements](#)
- [Sharding Advisor Schema Analysis Tool](#)

Centralized Backup and Restore of a Sharded Database

Oracle Sharding backup and recovery operations are centralized using new commands in the `GDSCTL` utility. You can define a backup policy for a sharded database as a whole and restore one or more shards, or the entire sharded database, to the same point in time. Configured backups are run automatically, and you can define a schedule to run backups during off-peak hours.

This feature streamlines backup and restore configuration and simplifies the overall management of backup policies for all of the databases in a sharded database topology. In earlier releases, you had to manually configure backup policies for each shard and the shard catalog database. Some of this work could be done with Oracle Enterprise Manager, but there was no way to orchestrate a complete restore of a sharded database such that all shards and the shard catalog are restored to the same point in time.

Related Topics

- [Oracle® Database Using Oracle Sharding](#)

Create a Sharded Database from Multiple Existing Databases (Federated Sharding)

Convert a set of existing databases running the same application into a sharded database, without modifying the database schemas or the application. The databases can be geographically distributed and can have some differences in their individual schemas.

You can more easily issue queries across multiple independent databases running the same application when they are combined into a sharded database.

Related Topics

- [Oracle® Database Using Oracle Sharding](#)

Multi-Shard Query, Data Loading, and DML Enhancements

If any shards are unavailable during query execution, then the enhanced multi-shard query attempts to find alternate shards to operate on, and the query resumes without issuing a failure condition. Bulk data loading and DML can operate on multiple shards simultaneously.

Multi-shard queries are more fault-tolerant. Bulk data loading and DML operations can occur across all shards simultaneously, making these operations much faster.

Related Topics

- [Oracle® Database Using Oracle Sharding](#)

Sharding Advisor Schema Analysis Tool

Sharding Advisor is a standalone command-line tool that helps you redesign a database schema so that you can efficiently migrate an existing, non-sharded Oracle Database to an Oracle sharding environment. Sharding Advisor analyzes your existing database schema and produces a ranked list of possible sharded database designs.

Using the Sharding Advisor recommendations, you can experience a smoother, faster migration to Oracle Sharding. Sharding Advisor analysis provides you with the information you need to:

- Maximize availability and scalability
- Maximize query workload performance
- Minimize the amount of duplicated data on each shard

Related Topics

- [Oracle® Database Using Oracle Sharding](#)

Transactional Event Queues (TEQs)

Transactional Event Queues (TEQ) is the new name for AQ Sharded Queues which were introduced in Database 12c and are further enhanced in Database 19c. All features of AQ Sharded Queues are now in TEQ, plus some new ones.

- [Advanced Queuing Support for JSON Data Type](#)
- [Advanced Queuing: Kafka Java Client for Transactional Event Queues](#)
- [Advanced Queuing: PL/SQL Enqueue and Dequeue Support for JMS Payload in Transactional Event Queues](#)
- [Advanced Queuing: PL/SQL Enqueue and Dequeue Support for non-JMS Payload in Transactional Event Queues](#)
- [Advanced Queuing: Simplified Metadata and Schema in Transactional Event Queues](#)
- [Advanced Queuing: Transactional Event Queues for Performance and Scalability](#)

Advanced Queuing Support for JSON Data Type

Oracle Database Advanced Queuing now supports JSON data type.

Many client applications and microservices which use Advanced Queuing for messaging have better performance if they use the JSON data type to handle JavaScript Object Notation (JSON) messages.

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Advanced Queuing: Kafka Java Client for Transactional Event Queues

Kafka Java Client for Transactional Event Queues (TEQ) enables Kafka application compatibility with Oracle database. This provides easy migration of Kafka applications to TEQ.

Customers don't have to manage a separate Kafka infrastructure, and this feature simplifies the event-driven application architectures with an Oracle converged database that now includes events data.

Starting from this release, Kafka Java APIs can connect to Oracle database server and use Transactional Event Queues (TEQ) as a messaging platform. Developers can migrate an existing Java application that uses Kafka to the Oracle database. A client side library allows Kafka applications to connect to Oracle database instead of Kafka cluster and use TEQ messaging platform transparently.

Related Topics

- [Oracle® Database Transactional Event Queues and Advanced Queuing User's Guide](#)

Advanced Queuing: PL/SQL Enqueue and Dequeue Support for JMS Payload in Transactional Event Queues

PL/SQL APIs perform enqueue and dequeue operations for Java Message Service (JMS) payload in Transactional Event Queues. Similarly, the PL/SQL Array APIs are exposed to Transactional Event Queues JMS users. Since JMS supports heterogeneous messages in a single JMS destination, dequeue gets one of the five JMS message types back, but cannot predict what is the type of the next message received. Therefore, it can run into application errors with PL/SQL complaining about type mismatch. Oracle suggests that the application always dequeue from Transactional Event Queues using the generic type `AQ$_JMS_MESSAGE`.

Customers can use PL/SQL APIs to enqueue and dequeue JMS payloads in Transactional Event Queues to avoid client-server round trips.

Related Topics

- [Oracle® Database Transactional Event Queues and Advanced Queuing User's Guide](#)

Advanced Queuing: PL/SQL Enqueue and Dequeue Support for non-JMS Payload in Transactional Event Queues

PL/SQL APIs can now perform enqueue and dequeue operations for ADT and RAW payloads in Transactional Event Queues. Similarly, the PL/SQL array APIs are exposed to Transactional Event Queue users.

ADT payloads are important because they allow you to have different queue payloads required by applications with all the benefits of strong type checking.

Related Topics

- [Oracle® Database Transactional Event Queues and Advanced Queuing User's Guide](#)

Advanced Queuing: Simplified Metadata and Schema in Transactional Event Queues

Transactional Event Queues have fewer tables than AQ and implement multiple memory optimizations for higher throughput. Customers will see higher message throughput just by switching from AQ to Transactional Event Queues.

This feature provides improvement in performance, scalability, and manageability.

Related Topics

- [Oracle® Database Transactional Event Queues and Advanced Queuing User's Guide](#)

Advanced Queuing: Transactional Event Queues for Performance and Scalability

Oracle Transactional Event Queues have their queue tables partitioned into multiple Event Streams, which are distributed across multiple RAC nodes for high throughput messaging and streaming of events.

Partitioned tables form part of the foundation to scale and increase performance of Transactional Event Queues, especially on Oracle RAC or Exadata.

Related Topics

- [Oracle® Database Transactional Event Queues and Advanced Queuing User's Guide](#)

Security Solutions

- [Advanced Security](#)
- [Database Vault](#)
- [Security](#)

Advanced Security

- Ability to Control Heartbeats in United Mode and Isolated Mode PDBs
- Ability to Set the Default Tablespace Encryption Algorithm
- Enhanced Database Availability with Zero Downtime to Switch Over to an Updated PKCS#11 Library
- Improved Performance with Large Numbers of TDE Keys in Wallets or Oracle Key Vault
- Sharing of TDE Master Encryption Key Across Oracle Processes

Ability to Control Heartbeats in United Mode and Isolated Mode PDBs

You now can control the size of the batch of heartbeats that use Oracle Key Vault or OCI KMS (OCI Vault) for centralized key management.

The `HEARTBEAT_BATCH_SIZE` initialization parameter, new with this release, enables you to set the heartbeat batch size. The duration of the heartbeat period defaults to 3 seconds.

This enhancement benefits the situation where you have a very large deployment of PDBs (for example, 1000) that use Oracle Key Vault. By setting the heartbeat batch size, you can stagger the heartbeats across batches of PDBs to ensure that for each batch a heartbeat can be completed for each PDB within the batch during the heartbeat period, and also ensure that PDB keys can be reliably fetched from an Oracle Key Vault server and cached in the persistent state.

Related Topics

- [Oracle® Database Advanced Security Guide](#)

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`, `3DES168`, `ARIA128`, `ARIA192`, `ARIA256`, `SEED128`, and `GOST256`. 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

Overview

This practice shows how you can set the `TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM` dynamic parameter to define the default encryption algorithm for tablespace creation operations.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Set the default tablespace encryption algorithm

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

```

$ sqlplus / AS SYSDBA
Connected to:

SQL> SHOW PARAMETER TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM

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

```

- Change the tablespace encryption algorithm.

```

SQL> ALTER SYSTEM SET TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM=AES192;

System altered.

SQL> EXIT
$

```

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

```

$ sqlplus sys@PDB21 AS SYSDBA
Enter password:
Connected.
SQL> CREATE TABLESPACE tbstest DATAFILE SIZE 2M;

Tablespace created.

SQL>

```

Step 2 : Verify the tablespace encryption algorithm used

- Verify the result of the operation.

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

NAME	ENCRYPT
USERS	AES128
TBTEST	AES192

```
SQL> EXIT
```

```
$
```

Enhanced Database Availability with Zero Downtime to Switch Over to an Updated PKCS#11 Library

Starting with this release, Oracle Database can switch over to an updated PKCS#11 library without incurring any system downtime.

This release introduces a new `ADMINISTER KEY MANAGEMENT SWITCHOVER LIBRARY FOR ALL CONTAINERS` statement, which will enable an Oracle database to switch over from the PKCS#11 library that it is currently using to the updated PKCS#11 library.

In previous releases, it was necessary to completely shut down any TDE-enabled database that used an online TDE master encryption key in Oracle Key Vault before an update to the Oracle Key Vault endpoint software could be installed. After the updated PKCS#11 library was installed, the TDE-enabled database would need to be started up again. This complete shut down followed by a start up of the database instance was necessary because long-running background processes of the database instance could not be told to unload the earlier PKCS#11 library and load the updated one.

Starting with this release, to switch over the database server to use an updated endpoint shared PKCS#11 library, you execute the `ADMINISTER KEY MANAGEMENT SWITCHOVER TO LIBRARY 'fully_qualified_file_name_of_library' FOR ALL CONTAINERS;` statement to initiate the switch over operation.

Related Topics

- [Oracle® Database Advanced Security Guide](#)

Improved Performance with Large Numbers of TDE Keys in Wallets or Oracle Key Vault

Oracle Database 21c introduces improved performance for Transparent Data Encryption (TDE).

This enhancement enables faster wallet loading and key rotations in multitenant databases. It allows for faster execution of TDE administration tasks and PDB cloning operations.

Sharing of TDE Master Encryption Key Across Oracle Processes

Starting with this release, you can enable sharing of Transparent Data Encryption (TDE) master encryption keys across Oracle processes.

This enhancement allows TDE-enabled Oracle databases to have their TDE master encryption keys managed by the Oracle Cloud Infrastructure (OCI) key management service (KMS). To control this functionality, you set the `TDE_KEY_CACHE` initialization parameter.

Related Topics

- [Oracle® Database Advanced Security Guide](#)

Database Vault

- Ability to Prevent Local Oracle Database Vault Policies from Blocking Common Operations
- ADMINISTER KEY MANAGEMENT Statement Now Protected by Oracle Database Vault Command Rules
- DBA_DV_SIMULATION_LOG View Columns REALM_NAME and RULE_SET_NAME Now VARCHAR2 Data Type
- No Need to Disable Oracle Database Vault Before Upgrades
- Uninstalling and Installing Oracle Label Security and Oracle Database Vault Now Supported

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 `DVSY` or `CTXS` 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

Overview

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 such as `DVSYS` or `CTXSYS` and prevent them from functioning correctly. For the purposes of this practice, the `C##TEST1` custom schema is created in CDB root to show this feature.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Configure and enable Database Vault at the CDB and PDB levels

- Configure and enable Database Vault at the CDB root level and at the PDB level. The script creates the `HR.G_EMP` table in the root container and also the `HR.L_EMP` table in `PDB21`.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_DV.sh
$ ./setup_DV_CDB.sh
...
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 contai
ner=ALL;

Grant succeeded.

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

Grant succeeded.

SQL> EXIT
```

```
...
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

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

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

Connected to:

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

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

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

Connected to:
```

```

CONNECTED TO.

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

$

```

Step 2 : Test table data accessibility with no realm on common objects

- 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:

SQL> SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;

NAME                                STATU
-----                                -
DV_ALLOW_COMMON_OPERATION FALSE

SQL>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

Step 3: Test table data accessibility with a common regular or mandatory realm on common objects

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

```

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

PL/SQL procedure successfully completed.

SQL> BEGIN
  DEMS_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>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

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

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

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.1_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Drop the realm.

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

PL/SQL procedure successfully completed.

SQL>
```

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


```

SQL> BEGIN
DEMS_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
DEMS_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>

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

```

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

SQL>

```

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

```

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

SQL>

```

- o Connect to PDB21 as C##TEST1, the table common owner.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

- o Connect to PDB21 as C##TEST2, another common user.

```

SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

- o Drop the realm.

```

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

PL/SQL procedure successfully completed.

SQL>

```

Step 4 : Test table data accessibility on common objects with a PDB regular or mandatory realm

- Create a PDB regular realm on C##TEST1 tables in PDB21.

```

SQL> CONNECT sec_admin@PDB21
Enter 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>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.1_emp;
SELECT * FROM c##test1.1_emp
                *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

- Drop the realm.

```
SQL> CONNECT sec_admin@PDB21
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm')

PL/SQL procedure successfully completed.

SQL>
```

- Create a PDB mandatory realm on C##TEST1 tables in PDB21.

```

SQL> CONNECT sec_admin@PDB21
Enter 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>

```

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

```

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

NAME                SALARY
-----
EMP_GLOBAL          1000

SQL>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL          1000

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;
SELECT * FROM c##test1.l_emp
                        *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;
SELECT * FROM c##test1.l_emp
                        *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

- Drop the realm.

```

SQL> CONNECT sec_admin@PDB21
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm')

PL/SQL procedure successfully completed.

SQL>

```

Step 5 : Restrict local users from creating Oracle Database Vault controls on common objects

```

SQL> CONNECT c##sec_admin
Enter 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>

```

Step 6 : Test table data accessibility with a common regular or mandatory realm on common objects

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


```

SQL> CONNECT c##sec_admin
Enter 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>

```

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

```

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

NAME                SALARY
-----
EMP_GLOBAL          1000

SQL>

```

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

```

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

SQL>

```

- Connect to PDB21 as C##TEST1, the table common owner.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

- Connect to PDB21 as C##TEST2, another common user.

```

SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

- Drop the realm.

```

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

PL/SQL procedure successfully completed.

SQL>

```

- 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>

```

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

```

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

SQL>

```

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

```

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

SQL>

```

- o Connect to PDB21 as C##TEST1, the table common owner.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>

```

- o Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.1_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- o Drop the realm.

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

PL/SQL procedure successfully completed.

SQL>
```

Step 7 : Test table data accessibility on common objects with a PDB regular or mandatory realm

- Create a PDB regular realm on C##TEST1 tables in PDB21.

```

SQL> CONNECT sec_admin@PDB21
Enter 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 us
er's object or a larger scope role to a realm.

SQL>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.1_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Drop the realm.

```
SQL> CONNECT sec_admin@PDB21
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm1')

PL/SQL procedure successfully completed.

SQL>
```

- Create a PDB mandatory realm on C##TEST1 tables in PDB21.


```

SQL> CONNECT sec_admin@PDB21
Enter 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     => 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>

```

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

```

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

NAME           SALARY
-----
EMP_GLOBAL     1000

SQL>

```

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

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

NAME                SALARY
-----
EMP_GLOBAL           1000

SQL>
```

- Connect to PDB21 as C##TEST1, the table common owner.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Connect to PDB21 as C##TEST2, another common user.

```
SQL> CONNECT c##test2@PDB21
Enter password:
Connected.
SQL> SELECT * FROM c##test1.l_emp;

NAME                SALARY
-----
EMP_LOCAL            2000

SQL>
```

- Drop the realm.

```

SQL> CONNECT sec_admin@PDB21
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Test Realm1')

PL/SQL procedure successfully completed.

SQL> EXIT
$

```

Step 8 : Summary

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.

Step 9 : Disable Database Vault in both the PDB and the CDB root

- Run the `disable_DV.sh` script to disable Database Vault in both the PDB and the CDB root.

```
$ /home/oracle/labs/M104781GC10/disable_DV.sh
...
SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit

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

Connected to:

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit

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

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area 851439688 bytes
Fixed Size                9691208 bytes
Variable Size             423624704 bytes
Database Buffers          281018368 bytes
Redo Buffers              19664896 bytes
In-Memory Area            117440512 bytes
Database mounted.
Database opened.

SQL> exit

$
```

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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Configure and enable Database Vault at the CDB and PDB levels

- Configure and enable Database Vault at the CDB root level and at the PDB level. The script creates the C##TEST1 and C##TEST2 common users.

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

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

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 contai
ner=ALL;

Grant succeeded.

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

Grant succeeded.

SQL> EXIT
```

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

Connected to:

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

Grant succeeded.

...
Connected to:

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

$
```

Step 2 : Test CDB and PDB connections with no command rule on common users

- 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
Enter 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>

```

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

```

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

```

- Connect to PDB21 as C##TEST1.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL>

```

Step 3 : Test CDB and PDB connections with a command rule in CDB root on common users

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

```
SQL> CONNECT c##sec_admin
Enter 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>
```

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

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

- Connect to PDB21 as C##TEST1.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL>
```

- Drop the command rule.


```

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

PL/SQL procedure successfully completed.

SQL>

```

Step 4 : Test CDB and PDB connections with a command rule in the PDB on common users

- Create a command rule on C##TEST1 in PDB21.

```

SQL> CONNECT c##sec_admin@PDB21
Enter 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>

```

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

```

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

```

- Connect to PDB21 as C##TEST1.

```
SQL> CONNECT c##test1@PDB21
ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON

Warning: You are no longer connected to ORACLE.

SQL>
```

- Drop the command rule.

```
SQL> CONNECT c##sec_admin@PDB21
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE('C##TEST1',DBMS_MACUTL.G_SCOPE_LOCAL)

PL/SQL procedure successfully completed.

SQL>
```

Step 5 : Prevent local users from creating Oracle Database Vault controls on common users that prevent them from logging in to the PDB

- Connect to the CDB root as C##SEC_ADMIN and switch the behavior of DV_ALLOW_COMMON_OPERATION.

```

SQL> CONNECT c##sec_admin
Enter 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>

```



You can execute this procedure without including any parameter to achieve a TRUE result.

Step 6 : Test CDB and PDB connections with a command rule in the CDB root on common users

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

```

SQL> CONNECT c##sec_admin
Enter 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>

```

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

```

SQL> CONNECT c##test1
Enter 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>

```

- Connect to PDB21 as C##TEST1.

```

SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL>

```

- Drop the command rule.

```

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

PL/SQL procedure successfully completed.

SQL>

```

Step 7 : Test CDB and PDB connections with a command rule in a PDB on common users

- Create a command rule on C##TEST1 in PDB21.

```

SQL> CONNECT sec_admin@PDB21
Enter 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>

```

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

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

- Connect to PDB21 as C##TEST1.

```
SQL> CONNECT c##test1@PDB21
Enter password:
Connected.
SQL> EXIT
$
```

Step 8 : Summary

Database Vault not only blocks 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.

Step 9 : Disable Database Vault in both the PDB and the CDB root

```
$ /home/oracle/labs/M104781GC10/disable_DV.sh

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

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

Connected to:

SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit

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

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

Connected to:
```

```
SQL> exec DVSYS.DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL> exit

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

Connected to:

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit

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
$
```

ADMINISTER KEY MANAGEMENT Statement Now Protected by Oracle Database Vault Command Rules

You now can protect the `ADMINISTER KEY MANAGEMENT` statement with Oracle Database Vault command rules.

The `ADMINISTER KEY MANAGEMENT` statement manages Transparent Data Encryption (TDE) features.

Related Topics

- [Oracle® Database Vault Administrator's Guide](#)

DBA_DV_SIMULATION_LOG View Columns REALM_NAME and RULE_SET_NAME Now VARCHAR2 Data Type

Starting with this release, the `REALM_NAME` and `RULE_SET_NAME` columns will use the `VARCHAR2` data type instead of being in nested tables.

This enhancement enables multiple realm names and rule set names to be separated by a comma in a `VARCHAR2` data type instead of using a nested table in the columns. In the unlikely situation where you may have so many realms or rule set names protecting a single object in which the `VARCHAR2` data exceeds 4000 characters, Oracle Database Vault will truncate the list of realms or rule sets at 4000 characters in the column and if the full set is needed, it can be retrieved from the nested table in the `DVSY$.SIMULATION_LOG$` base table. Storing realm names and rule set names as a `VARCHAR2` data type makes it easier for you to read the realm name or rule set name in the simulation log. Most users only use a single realm or rule set to protect their sensitive data objects and even if they do use multiple realms or rule sets, it is easier to read data in a `VARCHAR2` data type rather than a nested table.

Related Topics

- [Oracle® Database Vault Administrator's Guide](#)

No Need to Disable Oracle Database Vault Before Upgrades

Starting with this release, you do not need to disable Oracle Database Vault in every container before upgrading from an earlier release to the current release.

You only need to grant the `DV_PATCH_ADMIN` role to `SYS` commonly before you perform the upgrade. After the upgrade is complete the Database Vault controls work as before. Then revoke the `DV_PATCH_ADMIN` role from `SYS` commonly.

Alternatively, you can explicitly disable Oracle Database Vault in all containers before the upgrade, and then after the upgrade, explicitly enable Oracle Database Vault in all the containers.

Related Topics

- [Oracle® Database Vault Administrator's Guide](#)

Uninstalling and Installing Oracle Label Security and Oracle Database Vault Now Supported

You now can install and uninstall Oracle Database Vault and Oracle Label Security in PDBs.

To install a feature into a PDB requires that the feature already be installed in the CDB root.

This enhancement enables you to configure your own databases with Oracle Label Security and Oracle Database Vault to meet your site's requirements.

[Practice: Uninstalling Oracle Database Vault](#)

This practice shows how to remove Oracle Database Vault from an Oracle Database installation, for PDBs (but not the CDB root) and Oracle RAC installations.

[Practice: Installing Oracle Database Vault](#)

This practice shows how to install or reinstall Oracle Database Vault in a CDB by using Database Configuration Assistant (DBCA).

Related Topics

- [Oracle® Database Vault Administrator's Guide](#)

Practice: Uninstalling Oracle Database Vault

Overview

This practice shows how to uninstall Oracle Database Vault from an Oracle Database installation for PDBs (but not the CDB root) and Oracle RAC installations.

The uninstallation process does not affect the initialization parameter settings, even those settings that were modified during the installation process, nor does it affect Oracle Label Security.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Ensure Database Vault is enabled before uninstalling

- Execute the shell script to configure Database Vault at the CDB level.

```

$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_DV.sh
...
SQL> INSERT INTO l_tab values(2);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT

$

```

- Connect to the CDB root as C##SEC_ADMIN to verify the status of Database Vault.

```

$ sqlplus c##sec_admin

Enter password:

SQL> SELECT * FROM DVSYS.DBA_DV_STATUS;

NAME                                STATUS
-----                                -
DV_CONFIGURE_STATUS                 TRUE
DV_ENABLE_STATUS                    TRUE
DV_APP_PROTECTION                   NOT CONFIGURED

SQL>

```

- Log in to PDB21 as SYS with the SYSDBA administrative privilege.

```

SQL> CONNECT sys@PDB21 AS SYSDBA
Enter password:
Connected.
SQL> SELECT * FROM DVSYS.DBA_DV_STATUS;

NAME                                STATUS
-----                                -
DV_CONFIGURE_STATUS                 TRUE
DV_ENABLE_STATUS                     TRUE
DV_APP_PROTECTION                    NOT CONFIGURED

SQL> SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';

PARAMETER                                VALUE    CON_ID
-----                                -
Oracle Database Vault                   TRUE          0

SQL>

```

- Log in to the CDB root to ensure that the recycle bin is disabled.

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW PARAMETER recyclebin

NAME                                TYPE          VALUE
-----                                -
--
recyclebin                           string        on
SQL>

```

If the recycle bin is on, then disable it.

```

SQL> ALTER SYSTEM SET RECYCLEBIN = OFF SCOPE=SPFILE;

System altered.

SQL>

```

Step 2 : Disable Database Vault at the PDB and CDB levels

- Connect to PDB21 as a user who has been granted the DV_OWNER or DV_ADMIN role, such as C##SEC_ADMIN.

```
SQL> CONNECT c##sec_admin@PDB21
Enter password:
Connected.
SQL>
```

- Disable Oracle Database Vault at the PDB level.

```
SQL> EXEC DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL>
```

Proceed in all PDBs.

- Close and reopen PDB21.

```
SQL> CONNECT sys@PDB21 AS SYSDBA
Enter password:
Connected.
SQL> SHUTDOWN
Pluggable Database closed.
SQL> STARTUP
Pluggable Database opened.
SQL> SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';

PARAMETER                                VALUE    CON_ID
-----
Oracle Database Vault                     FALSE    0

SQL>
```



Even if the `CON_ID` displays 0, the value for the Database Vault refers to the PDB you are connected to.

- What is the status of Database Vault in the CDB root?

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';

PARAMETER                                VALUE    CON_ID
-----
Oracle Database Vault                    TRUE     0

SQL>
```

- Disable Oracle Database Vault at the CDB level.

```
SQL> CONNECT c##sec_admin
Enter password:
Connected.
SQL> EXEC DBMS_MACADM.DISABLE_DV

PL/SQL procedure successfully completed.

SQL>
```

- 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 1426060208 bytes
Fixed Size                 9687984 bytes
Variable Size              436207616 bytes
Database Buffers          973078528 bytes
Redo Buffers               7086080 bytes
Database mounted.
Database opened.
SQL> SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';

PARAMETER                                VALUE    CON_ID
-----
Oracle Database Vault                    FALSE      0

SQL>

```

Step 3 : Remove Database Vault metadata at the PDB and CDB levels

- Run the `dvremov.sql` script to remove Oracle Database Vault related metadata.

```

SQL> @$ORACLE_HOME/rdbms/admin/dvremov.sql
Session altered.

DECLARE
*
ERROR at line 1:
ORA-48000: Cannot run dvremov.sql from CDB root when one or more PDBs are
closed.
ORA-06512: at line 17

$

```

- Reopen PDB21 before removing Database Vault from the CDB root.


```

$ sqlplus / AS SYSDBA

Connected to:

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> @$ORACLE_HOME/rdbms/admin/dvremov.sql
Session altered.

DECLARE
*
ERROR at line 1:
ORA-47993: Cannot run dvremov.sql from CDB root when DV is installed in one o
r
more PDBs.
ORA-06512: at line 32

Disconnected from Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 -
Development
Version 21.1.0.0.0
$ oerr ORA 47993
47993, 00000, "Cannot run dvremov.sql from CDB root when DV is installed in o
ne or more PDBs."
// *Cause: The Database Vault (DV) removal script was not allowed to be run f
rom the multitenant
//          container database (CDB) root when DV is installed in one or more
of the underlying
//          pluggable databases (PDBs).
// *Action: Run dvremov.sql on all PDBs before running it from CDB root.
$
SQL>

```

- Run the `dvremov.sql` script to remove Oracle Database Vault related metadata from PDB21 and from all PDBs.

```
$ sqlplus sys@PDB21 AS SYSDBA
Enter password:
Connected.
SQL> @$ORACLE_HOME/rdbms/admin/dvremov.sql
Session altered.

PL/SQL procedure successfully completed.
...
User dropped.
...
Role dropped.

PL/SQL procedure successfully completed.
...
Grant succeeded.

PL/SQL procedure successfully completed.

Noaudit succeeded.
...
Commit complete.

PL/SQL procedure successfully completed.

Session altered.

SQL>
```

- Now remove Oracle Database Vault related metadata from the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> @$ORACLE_HOME/rdbms/admin/dvremov.sql
Session altered.

PL/SQL procedure successfully completed.
...
Commit complete.

PL/SQL procedure successfully completed.
...
Noaudit succeeded.

Commit complete.

PL/SQL procedure successfully completed.

Session altered.

SQL> EXIT
$
```

Practice: Installing Oracle Database Vault

Overview

This practice shows how to install or reinstall Oracle Database Vault in a CDB by using Database Configuration Assistant (DBCA).

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Ensure Database Vault is not installed

Before starting the reinstallation of Database Vault, review [Practice: Uninstalling Oracle Database Vault](#).

Step 2: Use DBCA to reinstall or install Database Vault in the CDB

- Before running the command, replace the password in the command below for both the DV owner and the DV account manager. Ensure that the DV owner and DV account manager accounts do not exist in the CDB root.

```

$ $ORACLE_HOME/bin/dbca -silent -configureDatabase -sourceDB CDB21 -dvConfiguration true -olsConfiguration true -dvUserName c##dvo -dvUserPassword password -dvAccountManagerName c##dvmgr -dvAccountManagerPassword password
Enter password for the TDE wallet: password

[WARNING] [DBT-16002] The database will be restarted in order to configure the chosen options.
Prepare for db operation
22% complete
Preparing to Configure Database
24% complete
29% complete
38% complete
40% complete
44% complete
Oracle Database Vault
89% complete
Completing Database Configuration
100% complete
The database configuration has completed successfully.
Look at the log file "/u01/app/oracle/cfgtoollogs/dbca/CDB21/CDB212.log" for further details.
$

```

If you use a password for the Data Vault owner or Data Vault manager such as WElcome123## with consecutive repeating characters, the command errors out immediately.

```

$ [FATAL] [DBT-12010] The Data Vault Owner password cannot contain any consecutive repeating characters.
[FATAL] [DBT-12010] The Data Vault Manager password cannot contain any consecutive repeating characters.
$

```

If you use a password for the Data Vault owner or Data Vault manager such as WElcome123#4, the command errors out only at the end of the dbca operation.

```
$ $ $ORACLE_HOME/bin/dbca -silent -configureDatabase -sourceDB CDB21 -dvConfiguration true -olsConfiguration true -dvUserName c##dvo -dvUserPassword WELcome123#4 -dvAccountManagerName c##dvacctmgr -dvAccountManagerPassword WELcome123#4
```

Enter password for the TDE wallet: *password*

```
[WARNING] [DBT-16002] The database will be restarted in order to configure the chosen options.
```

```
Prepare for db operation
```

```
22% complete
```

```
Preparing to Configure Database
```

```
24% complete
```

```
29% complete
```

```
38% complete
```

```
40% complete
```

```
44% complete
```

```
Oracle Database Vault
```

```
89% complete
```

```
Completing Database Configuration
```

```
[WARNING] ORA-28003: password verification for the specified password failed
```

```
ORA-20000: password must contain 2 or more special characters
```

```
[WARNING] ORA-28003: password verification for the specified password failed
```

```
ORA-20000: password must contain 2 or more special characters
```

```
[WARNING] ORA-01918: user 'c##dvo' does not exist
```

```
ORA-06512: at "DVSYS.DBMS_MACUTL", line 34
```

```
ORA-06512: at "DVSYS.DBMS_MACUTL", line 399
```

```
ORA-06512: at "DVSYS.CONFIGURE_DV_INTERNAL", line 46
```

```
ORA-06512: at "SYS.CONFIGURE_DV", line 173
```

```
ORA-06512: at line 1
```

```
[WARNING] ORA-01017: invalid username/password; logon denied
```

```
100% complete
```

```
The database configuration has completed successfully.
```

```
Look at the log file "/u01/app/oracle/cfgtoollogs/dbca/CDB21_fra1xn/CDB21_fra1xn0.log" for further details.
```

```
$
```

- Connect to the CDB root as C##DVO to verify the status of Database Vault.

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

SQL> SELECT * FROM DVSYS.DBA_DV_STATUS;

NAME                                STATUS
-----                                -
DV_CONFIGURE_STATUS                 TRUE
DV_ENABLE_STATUS                     TRUE
DV_APP_PROTECTION                     NOT CONFIGURED

SQL> SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';

PARAMETER                                VALUE    CON_ID
-----                                -
Oracle Database Vault                    TRUE         0

SQL> EXIT
$
```

Security

- [Oracle Blockchain Table](#)
- [Immutable Tables](#)
- [Authentication and Authorization](#)
- [Encryption](#)
- [Audit](#)

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 and 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 can be used to implement blockchain applications where the participants trust the Oracle Database provider, but want means to verify that their data hasn't been tampered with. The participants are different database users who trust the Oracle Database provider to maintain a verifiable, tamper-resistant blockchain of transactions. All participants must have privileges to insert data into the blockchain table. The contents of the blockchain table are defined and managed by the application, with a few added metadata fields maintained by Oracle Database. By leveraging a trusted provider with verifiable crypto-secure data management practices, such applications can avoid the distributed consensus requirements. This provides most of the protection of the distributed peer-to-peer blockchains, but with much higher throughput and lower transaction latency compared to peer-to-peer blockchains using distributed consensus.

[Details: Oracle Blockchain Table](#)

This page 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](#)

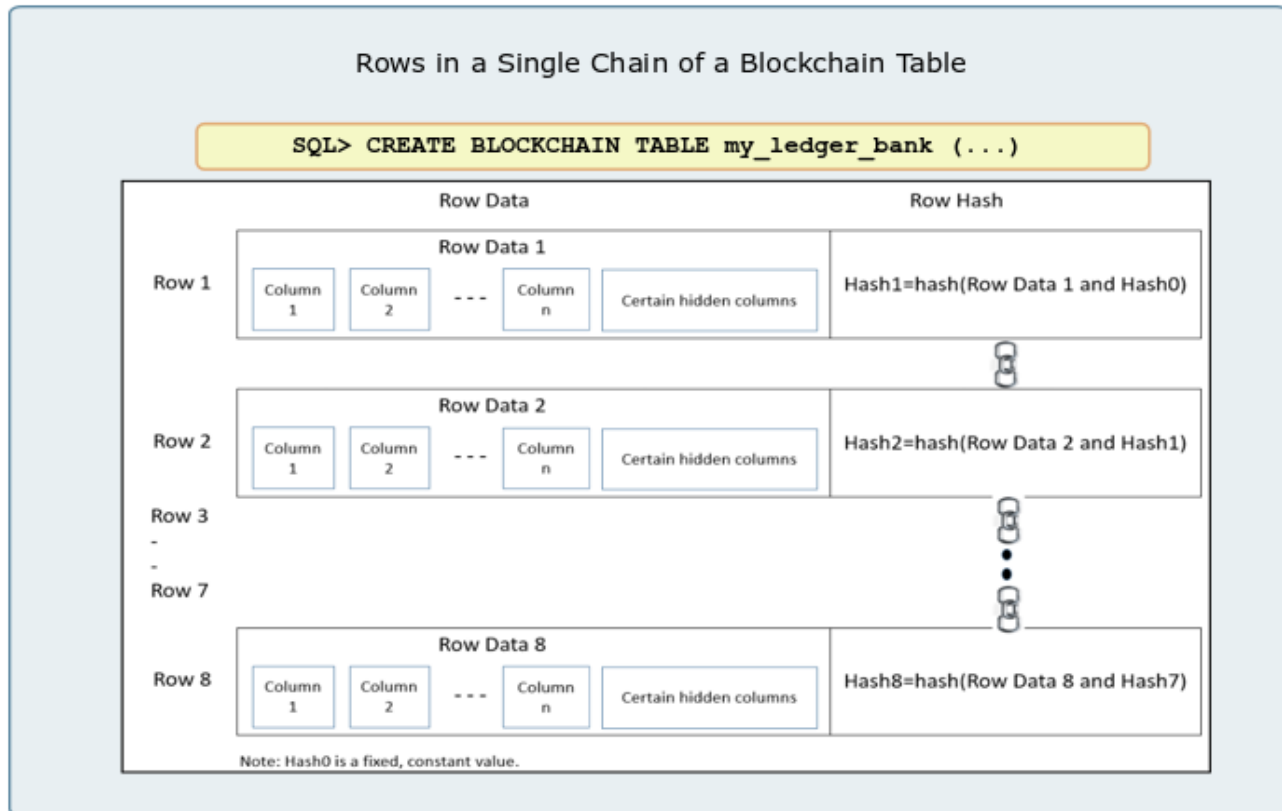
This practice shows how to create, alter and drop Oracle blockchain tables.

Related Topics

- [Oracle® Database Administrator's Guide](#)

Details: Oracle Blockchain Table

This page 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.

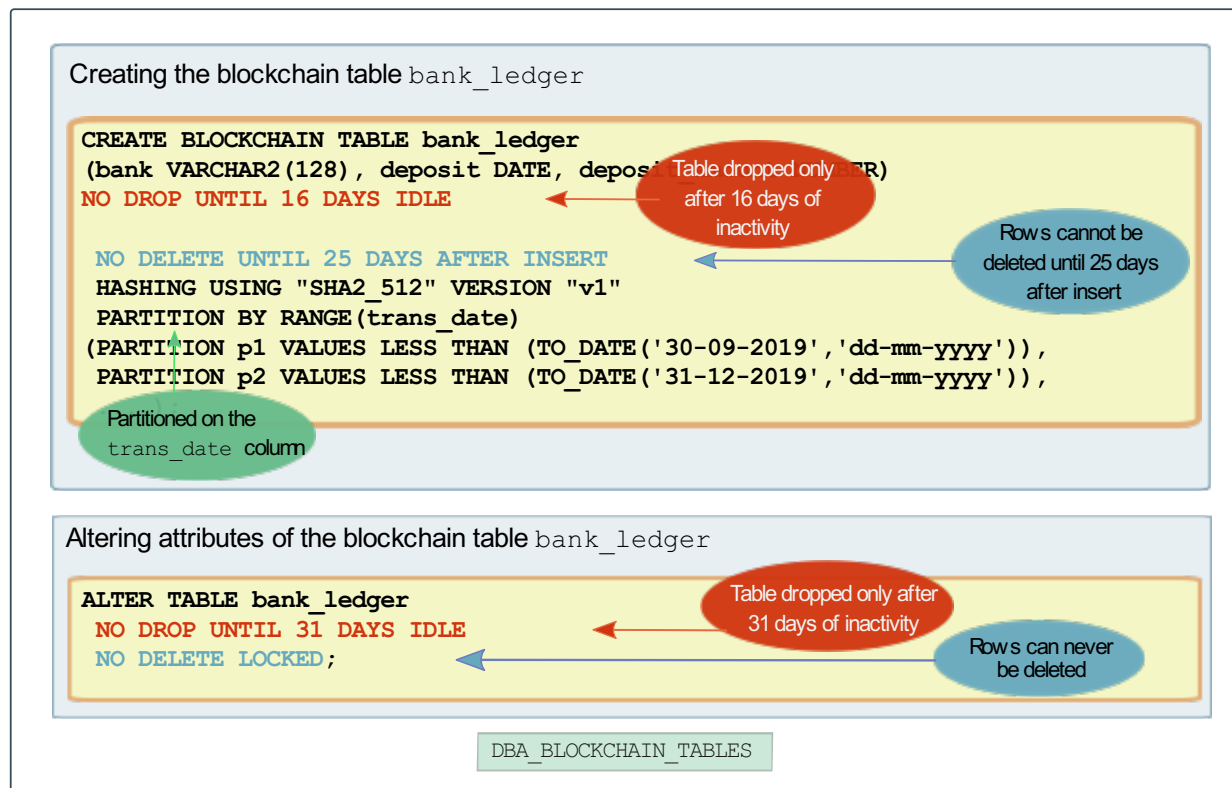
Blockchain tables are insert-only tables that organize rows into a number of chains. Each row, except the first row in the chain, is chained to the previous row.

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.

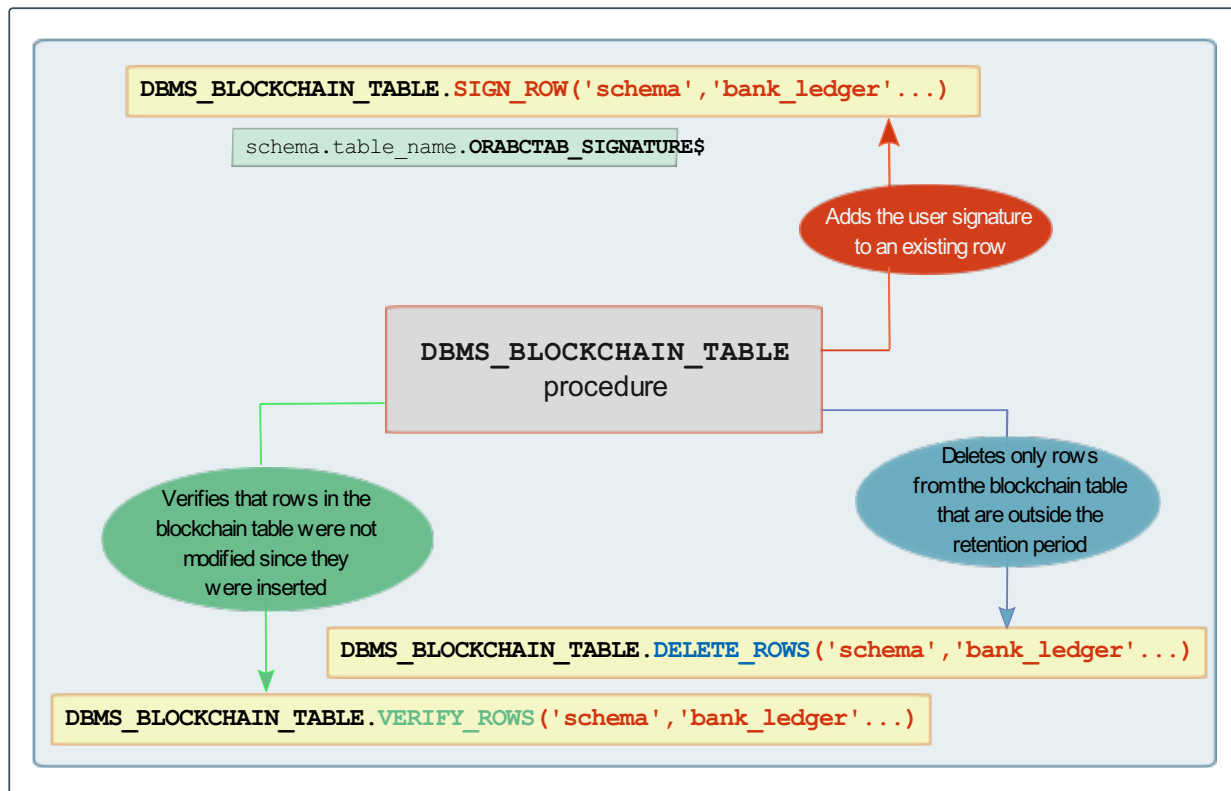
Consider the following benefits of using blockchain tables:

- 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 and 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 `DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS` procedure 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 `DBMS_BLOCKCHAIN_TABLE.DELETE_ROWS` procedure deletes all rows or rows that were created before a specified date.

Practice: Managing Blockchain Tables and Rows

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create the blockchain table

- Create the AUDITOR user, owner of the blockchain table.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_user.sh
...
specify password for HR as parameter 1:

specify default tablespace 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>

Connected to:

SQL> DROP USER auditor CASCADE;
DROP USER auditor CASCADE
      *
ERROR at line 1:
ORA-01918: user 'AUDITOR' does not exist

SQL> DROP TABLESPACE ledgertbs INCLUDING CONTENTS AND DATAFILES cascade const
raints;
DROP TABLESPACE ledgertbs INCLUDING CONTENTS AND DATAFILES cascade constraint
s
      *
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

$

```

- Create the blockchain table named `AUDITOR.LEDGER_EMP` that will maintain a tamper-resistant ledger of current and historical transactions about `HR.EMPLOYEES` in `PDB21`. Rows can never be deleted in the `AUDITOR.LEDGER_EMP` blockchain table. The blockchain table can be dropped only after 31 days of inactivity.

```

$ sqlplus auditor@PDB21

Enter 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

SQL>

```



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>
```

- 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>
```

- Show the description of the table.

```
SQL> DESC ledger_emp
```

Name	Null?	Type
-----	---	-----

EMPLOYEE_ID		NUMBER
SALARY		NUMBER

```
SQL>
```



Observe that the description displays only the visible columns.

- Use the `USER_TAB_COLS` view to display all internal column names used to store internal information such as the user number and the user 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>

```

Step 2 : Insert rows into the blockchain table

- Insert a first row into the blockchain table.

```

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

1 row created.

SQL> COMMIT;

Commit complete.

SQL>

```

- 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
5812238B734B019EE553FF8A7FF573A14CFA1076AB312517047368D600984CFAB001FA1FF2C98
B13
9AB03DDCCF8F6C14ADF16FFD678756572F102D43420E69B3

SQL>

```

- Connect as HR and insert a row into the blockchain table as 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>

```

- Connect as HR and insert a new row.


```

SQL> CONNECT hr@PDB21
Enter password:
Connected.
SQL> INSERT INTO auditor.ledger_emp VALUES (106,24000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL>

```

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

```

SQL> CONNECT auditor@PDB21
Enter password:
Connected.
SQL> SELECT ORABCTAB_CHAIN_ID$ "Chain ID", ORABCTAB_SEQ_NUM$ "Seq Num",
           to_char(ORABCTAB_CREATION_TIME$, 'dd-Mon-YYYY hh-mi') "Chain da
           te",
           ORABCTAB_USER_NUMBER$ "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	5812238B734B019EE553FF8A7FF573A14CFA1076AB312517047368D600984CFAB001FA1FF2C98B13	106	12000
14	2	06-Apr-2020 12-28	118	BBCDACC41B489DFBD8E28244841411937BD716F987BE750146572C555311E377D6DBA28D392C61E7	106	24000

```

SQL>

```



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

Step 3 : Delete rows from the blockchain table

- Delete the row inserted by HR.

```
SQL> DELETE FROM ledger_emp WHERE ORABCTAB_USER_NUMBER$ = 119;
DELETE FROM ledger_emp WHERE ORABCTAB_USER_NUMBER$ = 119
      *
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, NU
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.

- 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>
```

- 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 can't 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.

Step 4 : Drop the blockchain table

- 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>
```



Observe that the error message is slightly different. The error message from the `TRUNCATE TABLE` command explained that the operation was not possible on a blockchain table. The current error message explains that the `DROP TABLE` command is not possible on the `LEDGER_EMP` table.

The blockchain table was created so that it cannot be dropped before 31 days of inactivity.

- 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 dropping and removing any historical information that needs to be kept for security purposes.

Step 5 : Check the validity of rows in the blockchain table

- Create another blockchain table named `AUDITOR.LEDGER_TEST`. Rows cannot be deleted until 5 days after they were inserted. The blockchain table can be dropped only after 1 day of inactivity.

```
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 V
ARCHAR2(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>
```

- Connect as `HR` and insert a row into the blockchain table as 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>
```

- Connect as HR and insert a new row.

```
SQL> CONNECT hr@PDB21
Enter password:
Connected.
SQL> INSERT INTO auditor.ledger_test VALUES (1,'A1');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

- Connect as AUDITOR and display the row that was inserted.

```
SQL> CONNECT auditor@PDB21
Enter password:
Connected.
SQL> SELECT * FROM auditor.ledger_test;

          ID LA
----- --
          1 A1

SQL>
```

- Verify that the content of the rows is still valid. Use the `DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS` procedure.

```
SQL> CONNECT auditor@PDB21
Enter 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_IN
ST_ID$=instance_id;
        DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS('AUDITOR','LEDGER_TEST', NULL, NULL, in
stance_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
$
```

Immutable Tables

Immutable tables are insert-only tables in which existing data cannot be modified. Deleting rows is either prohibited or restricted based on the insertion time of the rows.

Immutable tables protect data against unauthorized modification by insiders. This includes database administrators or compromised users who have access to insider credentials. Immutable tables also prevent accidental data modification that may be caused by human error.

[Details: Immutable Tables](#)

This page provides more detailed information about immutable tables.

[Practice: Immutable Tables](#)

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

Related Topics

- [Oracle® Database Administrator's Guide](#)

Details: Immutable Tables

Immutable tables are insert-only tables that protect against unauthorized data modification or deletion.

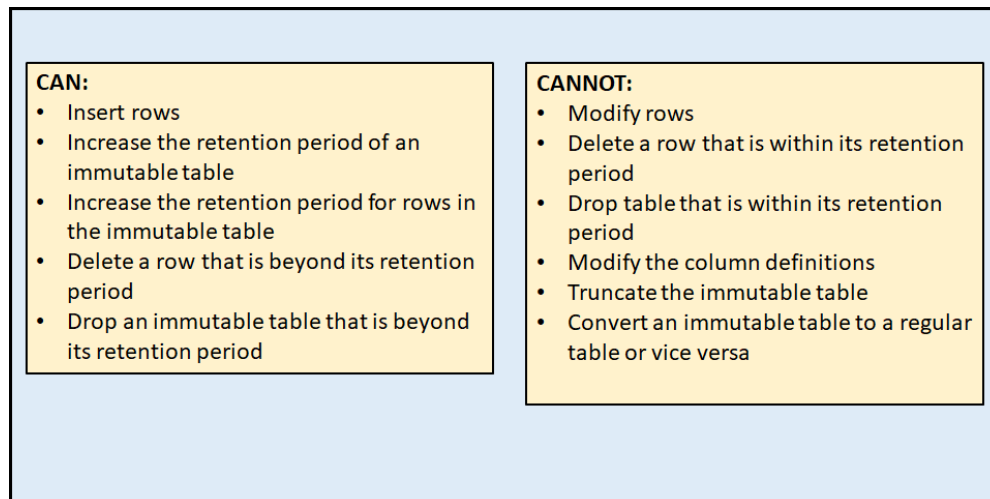
As part of the immutable table definition, you must specify a retention period for the immutable table and for rows within the immutable table. When an immutable table, or a row, is beyond its specified retention period, it is considered obsolete and is eligible for removal from the database.

Note that the `COMPATIBLE` initialization parameter must be 19.11.0.0 or higher to create immutable tables.

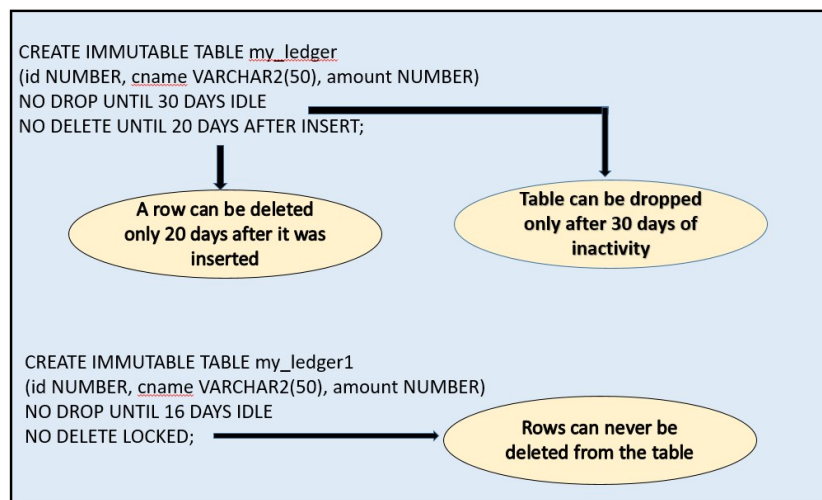
Why Immutable Tables?

- To prevent unauthorized modification of data by compromised insiders who have access to user credentials
- To prevent accidental modification of data caused by human error

Actions with Immutable Tables



Creating an Immutable Table



SQL Commands and PL/SQL Procedures for Immutable Tables

- `CREATE IMMUTABLE TABLE`
Creates the definition of the immutable table in the data dictionary
- `INSERT`
Adds a row to an immutable table
- `ALTER TABLE`
 - Increases the retention period of the immutable table
 - Increases the retention period of rows within the immutable table
- `DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWS` procedure
Deletes obsolete rows from the immutable table. A row is considered to be obsolete if it is beyond the set retention period.
- `DROP TABLE`
Drops an immutable table

Practice: Immutable Tables

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create an Immutable Table

- Create the `AUDITOR` user, the owner of the immutable table.
Skip this step if you have already created this user as part of the practice on managing blockchain tables.
- Connect as the `AUDITOR` user.
- Create an immutable table named `imm_ledger` in the `AUDITOR` schema of pluggable database `PDB21`. This immutable table can be dropped only after 20 days of inactivity. A row in the immutable table can be deleted only 30 days after it was inserted.

```
SQL> CREATE IMMUTABLE TABLE imm_ledger
 2  (
 3    tran_id NUMBER,
 4    tran_source VARCHAR2(50),
 5    tran_destination VARCHAR2(50),
 6    amount NUMBER
 7  )
 8  NO DROP UNTIL 20 DAYS IDLE
 9  NO DELETE UNTIL 30 DAYS AFTER INSERT;
```

Table created.

- Verify the attributes set for the immutable table by querying the `USER_IMMUTABLE_TABLES` data dictionary view.

```
SQL> SELECT row_retention "Row Retention Period", row_retention_locked "Row R
 2  etention Locked?", table_inactivity_retention "Table Retention Period"
 3  FROM user_immutable_tables
 4  WHERE table_name LIKE 'IMM_LEDGER';
```

```
Row Retention Period Row Retention Locked? Table Retention Period
```


```
-----
```

30	NO	20
----	----	----

- Show the description of the immutable table

▼ Show the description of the immutable table.

```
SQL> desc imm_ledger
Name                                     Null?      Type
-----
---
TRAN_ID                                 NUMBER
TRAN_SOURCE                             VARCHAR2 (50)
TRAN_DESTINATION                         VARCHAR2 (50)
AMOUNT                                   NUMBER
```

 Observe that the description displays only the user-created columns.

- Display details about all the columns in the immutable table by querying the `USER_TAB_COLS` data dictionary view. This includes the system-generated, hidden columns that are populated when a row is inserted. Hidden column names begin with `"ORABCTAB_"` and end with a dollar sign.

```
SQL> COL "Data Length" FORMAT 9999
SQL> COL "Column Name" FORMAT A24
SQL> COL "Data Type" FORMAT A28
SQL> SET LINESIZE 200
SQL> SELECT internal_column_id "Column ID", SUBSTR(column_name,1,30) "Column
Name", SUBSTR(data_type,1,30) "Data Type", data_length "Data Type Length"
 2 FROM user_tab_cols
 3 WHERE table_name LIKE 'IMM_LEDGER'
 4 ORDER BY internal_column_id;
```

Column ID	Column Name	Data Type	Data Type Length
1	TRAN_ID	NUMBER	22
2	TRAN_SOURCE	VARCHAR2	50
3	TRAN_DESTINATION	VARCHAR2	50
4	AMOUNT	NUMBER	22

```

22      5 ORABCTAB_INST_ID$      NUMBER
22      6 ORABCTAB_CHAIN_ID$    NUMBER
22      7 ORABCTAB_SEQ_NUM$    NUMBER
13      8 ORABCTAB_CREATION_TIME$  TIMESTAMP(6) WITH TIME ZONE
22      9 ORABCTAB_USER_NUMBER$  NUMBER
2000   10 ORABCTAB_HASH$        RAW
2000   11 ORABCTAB_SIGNATURE$    RAW

Column ID Column Name          Data Type          Data Type Le
ngth
-----
-----
22      12 ORABCTAB_SIGNATURE_ALG$  NUMBER
16      13 ORABCTAB_SIGNATURE_CERT$  RAW
2000   14 ORABCTAB_SPARE$          RAW

14 rows selected.

```

 Note that not all the hidden columns are populated for immutable tables.

Step 2: Insert Rows into the Immutable Table

- Insert three rows into the immutable table and commit the transaction.

```

SQL> INSERT INTO imm_ledger VALUES (100, 'John', 'Jane', 540);

1 row created.

SQL> INSERT INTO imm_ledger VALUES (110, 'Mary', 'Tom', 658);

1 row created.

SQL> INSERT INTO imm_ledger VALUES (120, 'Sheila', 'Paul', 48);

1 row created.

SQL> commit;

Commit complete.

```

- Display the immutable table data, including all user-defined columns and one hidden column.

```

SQL> COL "Creation Timestamp" FORMAT A40
SQL> SELECT ORABCTAB_CREATION_TIME$ "Creation Timestamp", tran_id "Transaction ID", SUBSTR(tran_source,1,15) "From" , SUBSTR(tran_destination,1,15) "To", amount
  2 FROM imm_ledger;

```

Creation Timestamp	Transaction ID	From	To
19-MAR-21 05.21.21.867076 PM +00:00 540	100	John	Jane
19-MAR-21 05.22.27.292712 PM +00:00 658	110	Mary	Tom
19-MAR-21 05.22.42.052533 PM +00:00 48	120	Sheila	Paul

Step 3: Modify the Definition of the Immutable Table

- Modify the retention period of the immutable table `imm_ledger` by using the `ALTER TABLE` command. The retention period is increased from the current value of 20 days to 30 days.

```
SQL> ALTER TABLE imm_ledger NO DROP UNTIL 30 DAYS IDLE;  
  
Table altered.
```

- Modify the retention period for rows in the immutable table `imm_ledger` by using the `ALTER TABLE` command.

```
SQL> ALTER TABLE imm_ledger NO DELETE UNTIL 18 DAYS AFTER INSERT;  
ALTER TABLE imm_ledger NO DELETE UNTIL 18 DAYS AFTER INSERT  
*  
ERROR at line 1:  
ORA-05732: retention value cannot be lowered  
  
SQL> ALTER TABLE imm_ledger NO DELETE UNTIL 45 DAYS AFTER INSERT;  
  
Table altered.
```

The first `ALTER TABLE` command throws an error because an attempt is being made to reduce the retention period. You cannot reduce the retention period for rows.

The second `ALTER TABLE` command succeeds, and the new retention period for rows in the immutable table is 45 days.

Step 4 : Delete Obsolete Rows from an Immutable Table

- Use the `DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWS` PL/SQL procedure to delete obsolete rows from the immutable table. Any row that is beyond the set retention period is considered obsolete.

```

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2     number_rows NUMBER;
  3 BEGIN
  4     DBMS_IMMUTABLE_TABLE.DELETE_EXPIRED_ROWS('AUDITOR','IMM_LEDGER',NULL,
NUMBER_ROWS);
  5     DBMS_OUTPUT.PUT_LINE('Number of rows deleted = '|| NUMBER_ROWS);
  6 END;
  7 /
Number of rows deleted = 3

PL/SQL procedure successfully completed.

```



In Step 3, the row retention period was modified to 45 days. Because the above PL/SQL procedure is run 45 days after the rows were inserted, the three inserted rows are deleted. If you run this procedure within 45 days of the row creation, the output will show that zero rows were deleted.

Step 5: Drop the Immutable Table

- Use the `DROP TABLE` command to drop the immutable table. Note that the immutable table can be dropped only if it contains no rows or after it has not been modified for a period of time that is defined by its retention period. In this example, the obsolete rows were dropped in Step 4. The assumption is that no new rows were added after the first three rows were inserted.

```

SQL> DROP TABLE imm_ledger;

Table dropped.

```

- If new rows were added to the immutable table after the first three rows in Step 2, and those rows are not yet obsolete, your output will be as follows:

```

SQL> DROP TABLE imm_ledger;
drop table imm_ledger
*
ERROR at line 1:
ORA-05723: drop blockchain or immutable table imm_ledger not allowed

```

Authentication and Authorization

- [Ability to Specify the Location of the CMU Wallet and dsi.ora File with a Database Property](#)
- [Ability to Use Multiple Kerberos Principals with a Single Database Client](#)
- [Addition of USER_APPLICATION_ROLES Data Dictionary View](#)
- [Connect to Multiple Databases with Different Certificates from a Single Client](#)
- [Enterprise User Manager Support for Per-PDB Directory Service Connections](#)
- [Force Upgraded Password File to be Case Sensitive](#)
- [Gradual Database Password Rollover for Applications](#)
- [Minimum Password Length Enforcement for All PDBs](#)
- [New and Updated Password User Profiles for STIG and CIS](#)
- [Oracle Database Connections to Kerberos Servers Now Default to TCP](#)
- [Windows Authentication No Longer Uses NTLM by Default](#)

Ability to Specify the Location of the CMU Wallet and dsi.ora File with a Database Property

You now can specify the location of the centrally managed users (CMU) wallet and dsi.ora files for an individual PDB by using a database property on the PDB.

This enhancement enables a PDB administrator to specify a location to store these files rather than being limited by the sqlnet.ora WALLET_LOCATION parameter or having to use the default wallet locations. This feature works almost exactly as using WALLET_LOCATION or the default wallet location except that users with administrative privileges will not be able to start the database because directory objects are part of the database. To store the CMU wallet and dsi.ora files in the location path specified by a database directory object, you must set the CMU_WALLET database property to this directory object.

After you have set the CMU_WALLET database property to a directory object for an individual PDB, you should store the CMU wallet and dsi.ora files for this PDB in the location specified by the directory object, so that they can be accessed through the corresponding file systems.

Related Topics

- [Oracle® Database Advanced Security Guide](#)

Ability to Use Multiple Kerberos Principals with a Single Database Client

Starting with this release, when you configure Kerberos authentication for an Oracle Database client, you can specify multiple Kerberos principals with a single Oracle Database client.

To enable this functionality, you will need to create a separate credential cache for each user in the client and then use the connect string to specify the user.

In previous releases, you were restricted to one Kerberos principal for each Oracle Database client.

- [Oracle® Database Security Guide](#)

Addition of USER_APPLICATION_ROLES Data Dictionary View

Starting with this release, the `USER_APPLICATION_ROLES` data dictionary view provides a subset of roles that are available in `DBA_APPLICATION_ROLES` that only apply to the current user.

`USER_APPLICATION_ROLES` enables the current user to see all the application roles that have been granted to the user instead of the list of all possible application roles available in `DBA_APPLICATION_ROLES`.

Related Topics

- [Oracle® Database Security Guide](#)

Connect to Multiple Databases with Different Certificates from a Single Client

Starting with this release, you can configure database clients to maintain multiple Secure Sockets Layer (SSL) sessions using different SSL certificates.

This feature enables multi-threaded clients to use multiple wallets with different certificates for simultaneous SSL sessions.

This enhancement is especially useful for database clients simultaneously connecting to multiple cloud databases, each with a different certificate.

Related Topics

- [Oracle® Database Security Guide](#)

Enterprise User Manager Support for Per-PDB Directory

Service Connections

Enterprise User Security (EUS) is now able to connect a different directory service for each pluggable database (PDB). Previously in a multitenant database, all the containers connected to a single directory service.

This feature enables a large Oracle Database deployment to have different directory services. It also enables independent software vendors to allocate an individual PDB to connect to the appropriate customer's directory service.

Related Topics

- [Oracle® Database Enterprise User Security Administrator's Guide](#)

Force Upgraded Password File to be Case Sensitive

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.

[Force Upgraded Password File to be Case Sensitive](#)

This practice shows how the passwords in the password files in Oracle Database 21c 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 an Upgraded Password File to be Case Sensitive

Overview

This practice shows how the passwords in the password files in Oracle Database 21c are case-sensitive. In earlier Oracle Database releases, password files retain their original case-insensitive verifiers by default. The IGNORECASE parameter, to enable or disable password file case sensitivity, is removed. All passwords in new password files are case-sensitive.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1 : Display the password file format of CDB21

```
$ cd /u01/app/oracle/dbs/
$ ls -l orapwCDB21
-rw-r----- 1 oracle oinstall 2048 Dec 10 09:45 orapwCDB21
$ orapwd describe file=orapwCDB21
Password file Description : format=12
$
```

Step 2 : Change the SYS password and verify that the password is now case-sensitive

- Change the SYS user password in the password file.

```
$ orapwd file=$ORACLE_BASE/dbs/orapwCDB21 sys=Y force=Y format=12 ignorecase=
Y
Usage 1: orapwd file= force={y|n} asm={y|n}
        dbuniquename= format={12|12.2}
        delete={y|n} input_file=
        'sys={y | password | external()
          | global()}'
        'sysbackup={y | password | external()
          | global()}'
        'sysdg={y | password | external()
          | global()}'
        'syskm={y | password | external()
          | global()}'

Usage 2: orapwd describe file=

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),
dbunique name
    - 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',
        'dbunique name' 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 parameters that can be used in the command. IGNORECASE is not mentioned because it is now a deprecated parameter.

- Re-enter the command without the deprecated parameter.

```
$ orapwd file=/u01/app/oracle/dbs/orapwCDB21 sys=Y force=Y format=12
Enter password for SYS:
$
```

- Log on as SYS to CDB21.

```
$ sqlplus sys@CDB21 AS SYSDBA

Enter password: password_with_case-sensitiveness

Connected to:
SQL> CONNECT sys@CDB21 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>
```

- Display the list of the users.

```

SQL> CONNECT sys@CDB21 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
$

```

Gradual Database Password Rollover for Applications

Starting with this release, an application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, a database administrator can associate a profile having a non-zero limit for the `PASSWORD_ROLLOVER_TIME` password profile parameter, new with this release, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the `PASSWORD_ROLLOVER_TIME` limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

Before this enhancement, an administrator normally took the application down when the application database password was being rotated. This is because the password update requires changes on both the database and the application side. With the gradual database password rollover enhancement, the application can continue to use the older password until the new password is configured in the application.

In addition to the new clause `PASSWORD_ROLLOVER_TIME` in the `CREATE PROFILE` and `ALTER PROFILE` statements, the `ALTER USER` statement has a new clause, `EXPIRE PASSWORD ROLLOVER PERIOD`. The `ACCOUNT_STATUS` column of the `DBA_USERS` and `USER_USERS` data dictionary views have several new statuses indicating values to indicate rollover status.

Related Topics

- [Oracle® Database Security Guide](#)

Minimum Password Length Enforcement for All PDBs

Starting with this release, you can enforce a minimum password length on all PDBs by setting a mandatory profile in the CDB root.

The mandatory profile is a generic profile and can only have the `PASSWORD_VERIFY_FUNCTION` parameter in the `CREATE MANDATORY PROFILE` statement to define the password length.

This profile adds a minimum password length to the local profiles with which the PDB user is associated. Because a common user sets the mandatory profile in the CDB, a PDB administrator cannot remove the password length requirement from local profiles and allow users to set insecure short passwords.

To use this profile, you create a password verification function to define the password length, execute the `CREATE MANDATORY PROFILE` PL/SQL procedure to use the `PASSWORD_VERIFY_FUNCTION` function in a profile, and then set the `MANDATORY_USER_PROFILE` initialization parameter in the CDB root to apply it to all containers.

[Practice: Enforcing a Minimum Password Length on All PDBs](#)

This practice shows how to enforce CDB-wide the minimum password length for the database user accounts without restricting access to the database user profiles.

Related Topics

- [Oracle® Database Security Guide](#)

Practice: Enforcing a Minimum Password Length on All PDBs

Overview

This practice shows how to enforce CDB-wide, the minimum password length for database user accounts without restricting access to database user profiles.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create a mandatory profile in the CDB root

- Connect to the CDB root in CDB21.

```
$ sqlplus sys@CDB21 AS SYSDBA

Enter password:
Connected to:

SQL>
```

- Create the mandatory root profile. The mandatory root profile acts as an always-on user profile. Mandatory profile limits are enforced in addition to the existing limits from the profile which the user is assigned to. This creates a union effect in the sense that the password complexity verification script of the mandatory profile will be executed before the password complexity script from the profile of the user account (if any).

```

SQL> COL resource_name FORMAT A30
SQL> COL limit FORMAT A30
SQL> CREATE MANDATORY PROFILE c##prof_min_pass_len
        LIMIT PASSWORD_VERIFY_FUNCTION oral2c_stig_verify_function
        CONTAINER=ALL;

Profile created.

SQL> SELECT resource_name, limit, mandatory FROM cdb_profiles
        WHERE profile='C##PROF_MIN_PASS_LEN' AND resource_type='PASSWORD';

RESOURCE_NAME                                LIMIT                                          MAN
-----
PASSWORD_VERIFY_FUNCTION                    FROM ROOT                                     YES
FAILED_LOGIN_ATTEMPTS                       YES
PASSWORD_LIFE_TIME                           YES
PASSWORD_REUSE_TIME                          YES
PASSWORD_REUSE_MAX                           YES
PASSWORD_LOCK_TIME                           YES
PASSWORD_GRACE_TIME                          YES
INACTIVE_ACCOUNT_TIME                        YES
PASSWORD_ROLLOVER_TIME                       YES
PASSWORD_VERIFY_FUNCTION                    ORA12C_STIG_VERIFY_FUNCTION                 YES
FAILED_LOGIN_ATTEMPTS                       YES
PASSWORD_LIFE_TIME                           YES
PASSWORD_REUSE_TIME                          YES
PASSWORD_REUSE_MAX                           YES
PASSWORD_LOCK_TIME                           YES
PASSWORD_GRACE_TIME                          YES
INACTIVE_ACCOUNT_TIME                        YES
PASSWORD_ROLLOVER_TIME                       YES

18 rows selected.

SQL>

```

Step 2 : Set the MANDATORY_USER_PROFILE initialization parameter

- Set the initialization parameter to the profile name.


```
SQL> ALTER SYSTEM SET mandatory_user_profile=C##PROF_MIN_PASS_LEN;
```

```
System altered.
```

```
SQL> SHOW PARAMETER mandatory_user_profile
```

NAME	TYPE	VALUE
---	---	---
mandatory_user_profile	string	C##PROF_MIN_PASS_LEN

```
SQL>
```



The password verify function of the mandatory profile is envisioned to be always enforced from `CDB$ROOT`, which means that the password resource limit is always fetched and executed from `CDB$ROOT` and enforced on the PDBs in the entire CDB depending on the `MANDATORY_USER_PROFILE` initialization parameter.

Step 3: Replace the password verification function to enforce the minimum password length.

- Replace the password verification function.

```
SQL> CREATE OR REPLACE FUNCTION ora12c_stig_verify_function
      ( username VARCHAR2, password VARCHAR2, old_password VARCHAR2)
      RETURN BOOLEAN
      IS
      BEGIN
        -- mandatory verify function will always be evaluated regardle
ss of the
        -- password verify function that is associated to a particular
profile/user
        -- requires the minimum password length to be 10 characters
        IF NOT ora_complexity_check(password, chars => 10) THEN return
      (false);
      END IF;
      RETURN(true);
      END;
      /

Function created.

SQL>
```

Step 4: Test

- Create a new user named JOHN in PDB21.

```
SQL> CONNECT system@PDB21
Enter password:
Connected.
SQL> CREATE USER john IDENTIFIED BY pass;
CREATE USER john IDENTIFIED BY pass
*
ERROR at line 1:
ORA-28219: password verification failed for mandatory profile
ORA-20000: password length less than 10 characters

SQL> CREATE USER john IDENTIFIED BY password123;
CREATE USER john IDENTIFIED BY password123
*
ERROR at line 1:
ORA-28003: password verification for the specified password failed
ORA-20000: password must contain 2 or more uppercase characters

SQL> CREATE USER john IDENTIFIED BY PAssword123;
CREATE USER john IDENTIFIED BY PAssword123
*
ERROR at line 1:
ORA-28003: password verification for the specified password failed
ORA-20000: password must contain 2 or more special characters

SQL> CREATE USER john IDENTIFIED BY PAssword123##;

User created.

SQL> DROP USER john CASCADE;

User dropped.

SQL>
```

Step 5 : Reset the configuration

- Drop the mandatory profile in the root.

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> DROP PROFILE c##prof_min_pass_len;
DROP PROFILE c##prof_min_pass_len
*
ERROR at line 1:
ORA-02381: cannot drop C##PROF_MIN_PASS_LEN profile

SQL> !oerr ora 2381
02381, 00000, "cannot drop %s profile"
// *Cause: An attempt was made to drop PUBLIC_DEFAULT or a mandatory profile,
//          which is not allowed due to following restrictions:
//          * PUBLIC_DEFAULT profile can be dropped only when the database
//            is in migration mode.
//          * A mandatory profile can be dropped only if it is not set as
//            a
//            mandatory profile in root container (CDB$ROOT) of a multitenant
//            container database (CDB) or in a Pluggable Database (PDB).
// *Action: If you are trying to drop the PUBLIC_DEFAULT profile, try dropping
//            it during migration mode. If you are trying to drop a mandatory
//            profile, check the MANDATORY_USER_PROFILE system parameter setting
//            in the root container (CDB$ROOT) or in a Pluggable Database (PDB)
//            and retry the operation after resetting the MANDATORY_USER_PROFILE
//            system parameter by executing ALTER SYSTEM RESET DDL statement.

SQL>

```

- Reset the MANDATORY_USER_PROFILE initialization parameter first.

```
SQL> ALTER SYSTEM RESET mandatory_user_profile;

System altered.

SQL> SHOW PARAMETER mandatory_user_profile

NAME                                TYPE                                VALUE
-----                                -                                -
--
mandatory_user_profile              string                              C##PROF_MIN_PASS_LEN
SQL> DROP PROFILE c##prof_min_pass_len;
DROP PROFILE c##prof_min_pass_len
*
ERROR at line 1:
ORA-02381: cannot drop C##PROF_MIN_PASS_LEN profile

SQL>
```

- Restart the instance.

```

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

Total System Global Area 1426060208 bytes
Fixed Size                  9687984 bytes
Variable Size               419430400 bytes
Database Buffers            989855744 bytes
Redo Buffers                 7086080 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb21 OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER mandatory_user_profile

NAME                                TYPE                                VALUE
-----                                -                                -
--
mandatory_user_profile              string
SQL> DROP PROFILE c##prof_min_pass_len;

Profile dropped.

SQL> EXIT
$

```

New and Updated Password User Profiles for STIG and CIS

This release introduces a new user profile and an updated user profile to comply with STIG and CIS standards for password management.

The `ora_stig_profile` user profile has been updated with the latest Security Technical Implementation Guide's (STIG) guidelines, and the `ora_cis_profile`, new for this release, has the latest Center for Internet Security (CIS) guidelines for passwords. You can use these user profiles directly with their database users or use them as part of your own user profiles. Oracle keeps these profiles up to date to make it easier for you to implement password policies that meet STIG and CIS guidelines.

Related Topics

- [Oracle® Database Security Guide](#)

Oracle Database Connections to Kerberos Servers Now Default to TCP

In previous releases, Oracle Database connections to Kerberos servers defaulted to start with User Datagram Protocol (UDP).

If you prefer the Oracle Database connections to start with UDP connections, then you can modify the Kerberos `krb5.conf` configuration file to start the connections using UDP.

This enhancement speeds the authentication process for environments that have many Kerberos Key Distribution Center (KDC) servers.

Related Topics

- [Oracle® Database Security Guide](#)

Windows Authentication No Longer Uses NTLM by Default

For Microsoft Windows installations with `AUTHENTICATION_SERVICES=NTS`, starting with this release, the `SQLNET.NO_NTLM` parameter setting in the `sqlnet.ora` file will default to `TRUE`.

In previous releases, the default for this parameter was `FALSE`. `SQLNET.NO_NTLM` controls whether NTLM can be used with NTS authentication. A `TRUE` setting means that NTLM cannot be used in NTS authentication. Because NTLM does not normally provide mutual authentication and is hence less secure, a `TRUE` setting for `SQLNET.NO_NTLM` makes the database and client more secure.

The `SQLNET.NO_NTLM` parameter is used on both the server and the client. If you have upgraded a Microsoft Windows installation of an Oracle database or a client in which `SQLNET.NO_NTLM` had not been set, then its default will be `TRUE`.

Related Topics

- [Oracle® Database Security Guide](#)

Encryption

- [Updated Support for Micro Edition Suite \(MES\) for FIPS 140.2](#)
- [Support for DBMS_CRYPT0 Asymmetric Key Operations](#)

Updated Support for Micro Edition Suite (MES) for FIPS 140.2

Starting with this release, Oracle Database supports Micro Edition Suite (MES) version 4.5 for FIPS 140.2.

The Micro Edition Suite (MES) version 4.5 updates include four new CVEs in the RSA BSAFE MES library, support for the rules that FIPS 140.2 requires, and access to the updated NZ/ZT library from the Crypto Foundation.

This enhancement enables the Oracle Database FIPS 140.2 configuration to benefit from new features and security improvements available from the latest RSA BSAFE MES library.

Related Topics

- [Oracle® Database Security Guide](#)

Support for DBMS_CRYPT0 Asymmetric Key Operations

Starting with this release, the `DBMS_CRYPT0` PL/SQL package supports asymmetric key operations, in addition to the existing support for symmetric key operations.

To implement the support for asymmetric key operations, the following procedures have been added to the `DBMS_CRYPT0` package:

- `PKENCRYPT`
- `PKDECRYPT`
- `SIGN`
- `VERIFY`

Related Topics

- [Oracle® Database PL/SQL Packages and Types Reference](#)

Audit

- [Auditing for Oracle XML DB HTTP and FTP Services](#)
- [Predefined Unified Audit Policies for Security Technical Implementation Guide \(STIG\) Compliance](#)
- [SYSLOG Destination for Common Unified Audit Policies](#)
- [Unified Audit Policies Enforced on the Current User](#)
- [Unified Audit Policy Configuration Changes Effective Immediately](#)
- [Unified Auditing on an Editioned Object Now Applies to All Its Editions](#)

Auditing for Oracle XML DB HTTP and FTP Services

Starting with this release, you can create unified audit policies for database connections made using the database protocol servers for HTTP, HTTPS, and FTP.

A unified audit policy can track requests that have been made to servlets, such as those used by Oracle Enterprise Manager Express, Oracle Database Native Web Services, and HTTP and WebDAV operations that use Oracle XML DB Repository.

This enhancement enables you to track and monitor access to Oracle Database provided by the HTTP, HTTPS, and FTP protocols, including access by WebDAV clients.

Related Topics

- [Oracle® Database Security Guide](#)

Predefined Unified Audit Policies for Security Technical Implementation Guide (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 Guide (STIG) audit requirements.

Related Topics

- [Oracle® Database Security Guide](#)

Practice: Using Predefined Unified Audit Policies for STIG Compliance

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1 : View the predefined unified audit policies that are implemented

- Connect to PDB21 as SYSTEM.

```
$ sqlplus system@PDB21

Enter password:
Connected.

SQL>
```

- Determine which predefined unified audit policies are implemented. Observe the three new predefined unified audit policies implemented in Oracle Database 21c.

```
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_LOGON_FAILURES
ORA_LOGON_LOGOFF
ORA_RAS_POLICY_MGMT
ORA_RAS_SESSION_MGMT
ORA_SECURECONFIG
ORA_STIG_RECOMMENDATIONS

10 rows selected. POLICY_NAME


SQL>
```

- 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.

- 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_T_	AUDIT_OPTION_TYP	OBJE	OBJECT_NAME	OBJEC
--				
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      PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_JOB             PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_RLS             PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_REDACT         PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_TSDP_MANAGE    PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_TSDP_PROTECT   PACKA
GE
EXECUTE                       OBJECT ACTION    SYS   DBMS_NETWORK_ACL_ADMIN PACKA
GE

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.

- 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.

- 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 connections 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 Guide (STIG) requirements.

Step 2 : Enable all three audit policies for all users

- Audit all major actions that could damage the security and the smooth running of the database, all top level actions of privileged users on any object that could damage the security of the database, and all connections and disconnections in the database instance.

```
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

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1 : Create a common user

- 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 `CDB21`. The shell script creates a common user named `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
Connected to:

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
...
/usr/bin/ar cr /u01/app/oracle/product/21.0.0/dbhome_1/rdbms/lib/libknlopt.a
/u01/app/oracle/product/21.0.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/21.1.0/dbhome_1/bin

- Linking Oracle
rm -f /u01/app/oracle/product/21.0.0/dbhome_1/rdbms/lib/oracle
...
SQL> STARTUP
...
SQL> CREATE USER c##test IDENTIFIED BY password CONTAINER=ALL;

User created.

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

Grant succeeded.

SQL> EXIT
$
```

Step 2: Create a common and local audit policy

- Create the common audit policy at the CDB root in CDB21.

```

$ 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>

```

- Create the local audit policy at the PDB level in PDB21.

```

SQL> CONNECT system@PDB21
Enter password:
Connected.
SQL> CREATE AUDIT POLICY pol_pdb21 ACTIONS select;

Audit policy created.

SQL> AUDIT POLICY pol_pdb21;

Audit succeeded.

SQL>

```

- 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_PDB21	SELECT	NO

```

SQL>

```

Step 3: Configure the SYSLOG destination for common and local audit policies

- 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>

```

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

```

SQL> CONNECT sys@PDB21 AS SYSDBA
Enter 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 UNIFIED_AUDIT_COMMON_SYSTEMLOG is a CDB-level initialization parameter.

- Restart the database instance because 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 851440288 bytes
Fixed Size                 9691808 bytes
Variable Size             599785472 bytes
Database Buffers         222298112 bytes
Redo Buffers              19664896 bytes
Database mounted.
Database opened.
SQL>

SQL> ALTER PLUGGABLE DATABASE pdb21 OPEN;

Pluggable database altered.

SQL>
```

Step 4: Define the OS directories for the SYSLOG files

- 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 as `root`.

```
$ exit
logout
$ sudo su -
Last login: Tue Dec 15 00:00:07 UTC 2020
#
```

- 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
local11.warning                             /var/log/root_audit_records.log
...
#
```

- Restart the SYSLOG daemon.

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

Step 5 : Test

- In the oracle UNIX session, log on as C##TEST to the CDB root and perform a CREATE TABLE operation followed by an INSERT operation on the table.

```

SQL> CONNECT c##test
Enter password:
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.

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

1 row created.

SQL> COMMIT;

Commit complete.

SQL>

```

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

```

# grep -i 'Oracle Unified Audit' /var/log/root_common_audit_records.log
Nov 13 15:52:56 db21si journal: Oracle Unified Audit[23128]: LENGTH: '215' TYPE:
"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYID:"1" STMTID:"9
" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E704F40AD8A9"
#

```



If you query `AUDIT_ACTIONS`, you observe that the value in `ACTION` number 1 is `CREATE TABLE`.



If you query `V$CONTAINERS`, you observe that the `GUID` value corresponds to the CDB root.



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 (ACTION number 2 in `AUDIT_ACTIONS` view) 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.

- Check that syslog entries are created in `/var/log/root_audit_records.log` file because audit records for `INSERT` were generated because of the `POL_ROOT` local root audit policy.

```
# grep -i 'Oracle Unified Audit' /var/log/root_audit_records.log | grep TEST
Nov 13 15:52:56 db21si journal: Oracle Unified Audit[23128]: LENGTH: '215' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYD:"1" STMTID:"9"
DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1" RETCODE:"0" SCHEMA:"C##TEST" OB
JNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E04F40AD8A9"
Nov 13 15:53:02 db21si journal: Oracle Unified Audit[23128]: LENGTH: '216' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYD:"2" STMTID:"10
" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053704F40AD8A9"
Nov 13 15:53:05 db21si journal: Oracle Unified Audit[23128]: LENGTH: '216' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYD:"3" STMTID:"11
" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053704F40AD8A9"
#
```



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



If you query `V$CONTAINERS`, you observe that the `GUID` value corresponds to `PDB21`.

- Back in the `oracle` UNIX session, log on as `C##TEST` to `PDB21` and perform a `CREATE TABLE` operation, followed by an `INSERT` operation on the table.


```

SQL> CONNECT c##test@PDB21
Enter password:
Connected.
SQL> CREATE TABLE testpdb21 (id NUMBER, label VARCHAR2(10));

Table created.

SQL> INSERT INTO testpdb21 VALUES (1,'A');

1 row created.

SQL> INSERT INTO testpdb21 VALUES (2,'B');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> EXIT

$

```

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

```

# grep -i 'Oracle Unified Audit' /var/log/root_common_audit_records.log
Nov 13 15:52:56 db21si journal: Oracle Unified Audit[23128]: LENGTH: '215' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYID:"1" STMTID:"9
" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E704F40AD8A9"
Nov 13 16:01:47 db21si journal: Oracle Unified Audit[51696]: LENGTH: '219' TY
PE:"4" DBID:"3207694222" SESID:"502695322" CLIENTID:"" ENTRYID:"6" STMTID:"7"
DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1" RETCODE:"0" SCHEMA:"C##TEST" OB
JNAME:"TESTPDB21" PDB_GUID:"B3C43A538FDE55BEE0537167606449A3"
#

```



The second entry corresponds to the `CREATE TABLE` action audited commonly because the `POL_COMMON` common audit policy audits all `CREATE TABLE` statements in all containers and also in `PDB21`. No `INSERT` action is recorded in this log file because the `POL_ROOT` audit policy that audits `INSERT` statements, is created only locally in the `CDB` root and not commonly in all containers.

- Check whether syslog entries are created in the `/var/log/root_audit_records.log` file.

```
# grep -i 'Oracle Unified Audit' /var/log/root_audit_records.log | grep TEST
Nov 13 15:52:56 db21si journal: Oracle Unified Audit[23128]: LENGTH: '215' TYPE:
"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYID:"1" STMTID:"9"
" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"1" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E704F40AD8A9"
Nov 13 15:53:02 db21si journal: Oracle Unified Audit[23128]: LENGTH: '216' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYID:"2" STMTID:"1
0" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E704F40AD8A9"
Nov 13 15:53:05 db21si journal: Oracle Unified Audit[23128]: LENGTH: '216' TY
PE:"4" DBID:"2809789491" SESID:"2759083216" CLIENTID:"" ENTRYID:"3" STMTID:"1
1" DBUSER:"C##TEST" CURUSER:"C##TEST" ACTION:"2" RETCODE:"0" SCHEMA:"C##TEST"
OBJNAME:"TEST" PDB_GUID:"B3316DF8AB281563E053E704F40AD8A9"
# exit
logout
$
```



Although a local audit policy, POL_PDB21 in PDB21, audits INSERT actions, no audit record is written in the SYSLOG file because SYSLOG records only actions executed at the CDB level.

Step 6: Cleanup

- Back in the oracle UNIX session, execute the `/home/oracle/labs/M104781GC10/cleanup.sh` shell script to reset the SYSLOG destinations for both common and local unified audit policies and drop the policies in the CDB root and PDB21.

```
$ /home/oracle/labs/M104781GC10/cleanup.sh
...
SQL> ALTER SYSTEM SET UNIFIED_AUDIT_COMMON_SYSTEMLOG='' SCOPE=SPFILE;

System altered.

SQL> ALTER SYSTEM SET UNIFIED_AUDIT_SYSTEMLOG='' SCOPE=SPFILE;

System altered.

SQL> noaudit POLICY pol_common;

Noaudit succeeded.

SQL> drop AUDIT POLICY pol_common;

Audit Policy dropped.
```

```
SQL> exit
...
SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
...
Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area 851440088 bytes
Fixed Size                 9691608 bytes
Variable Size             570425344 bytes
Database Buffers         134217728 bytes
Redo Buffers              19664896 bytes
In-Memory Area           117440512 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> exit
...
SQL> noAUDIT POLICY pol_pdb21;

Noaudit succeeded.

SQL> drop AUDIT POLICY pol_pdb21;

Audit Policy dropped.

SQL> EXIT
$
```

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](#)

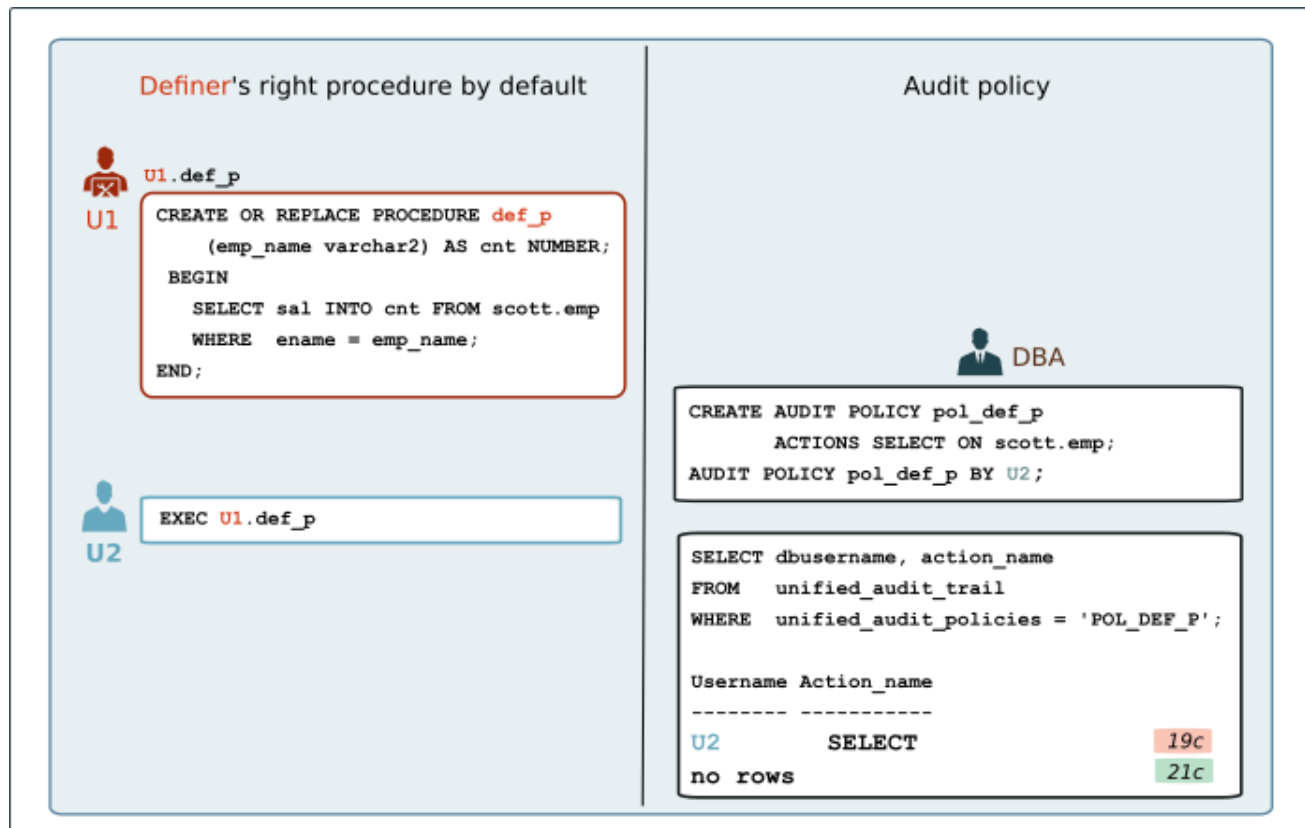
This practice shows how unified audit policies are enforced on the current user who executes the SQL statement.

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

Overview

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

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create the users and a procedure

- Execute the `/home/oracle/labs/M104781GC10/setup_audit_policies.sh` script to create users and a procedure for this practice.

```
$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_audit_policies.sh
...
Connected to:
```

```
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     ;
  8     dbms_output.put_line('Salary is : '||v_sal || ' for Employee ID: '||v
  9     _emp_id);
  8 END procemp;
  9 /

Procedure created.
```

```
SQL>
SQL> grant execute on u1.procomp to u2;

Grant succeeded.

SQL>
SQL> exit

$
```

Step 2 : Create and enable an audit policy

- In PDB21, create and enable an audit policy to audit any query on the HR.EMPLOYEES table executed by the login user named U2.

```
$ sqlplus system@PDB21

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

Enter password:

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

Audit policy created.

SQL> AUDIT POLICY pol_emp BY u2;

Audit succeeded.

SQL>
```

Step 3 : Test

- Connect to PDB21 as U2 and execute the U1.PROCEMP procedure.

```
SQL> CONNECT u2@PDB21
Enter password:
SQL> SET SERVEROUTPUT ON
SQL> EXECUTE u1.procomp(206)
Salary is : 8300 for Employee ID: 206

PL/SQL procedure successfully completed.

SQL>
```

- Display the values in `DBUSERNAME` (the login user) and `CURRENT_USER` (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
$
```



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 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.

Current sessions 19c

U1 CONNECT u1@PDB19

Audit policy

DBA CREATE AUDIT POLICY poll
ACTIONS SELECT ON scott.emp;
AUDIT POLICY poll BY ALL;

U1 SELECT * FROM scott.emp;

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

no rows selected

New sessions

U2 CONNECT u2@PDB19
SELECT * FROM scott.emp;

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

Username Action_name

U2 SELECT

Current sessions 21c

U1 CONNECT u1@PDB20

Audit policy

DBA CREATE AUDIT POLICY poll
ACTIONS SELECT ON scott.emp;
AUDIT POLICY poll BY ALL;

U1 SELECT * FROM scott.emp;

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

Username Action_name

U1 SELECT

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

Practice: Auditing Actions on Connected Sessions

Overview

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.

Before starting any new practice, refer to the [Practices Environment](#) recommendations.

Step 1: Create the local user in PDB21

```

$ cd /home/oracle/labs/M104781GC10
$ /home/oracle/labs/M104781GC10/setup_audit.sh
Copyright (c) 1982, 2020, Oracle. All rights reserved.

Connected to:
Oracle Database 20c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.2.0.0.0

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 20c Enterprise Edition Release 21.0.0.0.0 - Prod
uction
Version 21.2.0.0.0

LSNRCTL for Linux: Version 21.0.0.0.0 - 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/21.2.0/dbhome_1/rdbms/lib/libknlopt.a /u01/
app/oracle/product/21.2.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/21.2.0/dbhome_1/bin

- Linking Oracle
rm -f /u01/app/oracle/product/20.2.0/dbhome_1/rdbms/lib/oracle
...
LSNRCTL for Linux: Version 21.0.0.0.0 - Production on 20-MAR-2020 04:13:52

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

Starting /u01/app/oracle/product/21.2.0/dbhome_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 21.0.0.0.0 - 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.x
ml
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 21.0.0.0.0 - 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/listene
r.ora
Listener Log File   /u01/app/oracle/diag/tnslnsr/server/listener/alert/log.x
ml
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 21.0.0.0.0 - Production on Fri Mar 20 04:13:52 2020
Version 21.2.0.0.0

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

Connected to an idle instance.

SQL> STARTUP
...
SQL*Plus: Release 21.0.0.0.0 - Production on Mon Mar 9 05:09:56 2020
Version 21.2.0.0.0

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

Connected to:
Oracle Database 19c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.2.0.0.0

specify password for HR as parameter 1:

specify default tablespace 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

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

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

Connected to:
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
$
```

Step 2 : Create and enable an audit policy on the `HR.LOCATIONS` table

- Connect as U1 to PDB21.

```
$ sqlplus u1@PDB21
Copyright (c) 1982, 2020, Oracle. All rights reserved.

Enter password:

Connected to:
Oracle Database 20c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.2.0.0.0

SQL>
```

- In another terminal session, connect as SYSTEM to PDB21 and create and enable an audit policy to audit any SELECT on the HR.LOCATIONS table.
 - Verify that Unified Auditing is enabled.

```
$ sqlplus system@PDB21
Copyright (c) 1982, 2020, Oracle. All rights reserved.

Enter password:

Connected to:
Oracle Database 20c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.2.0.0.0

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

VALUE
-----
TRUE

SQL>
```

- 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>
```

Step 3: Test

- Back in the U1 session, select rows from the 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
$

```

- Is the query executed by U1 audited although the user has not reconnected? Switch back to the SYSTEM session.

```

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

Username Action_name
-----
U1          SELECT

SQL> EXIT
$

```

Unified Auditing on an Editioned Object Now Applies to All Its Editions

Unified audit policies, if configured for auditing actions on an editioned object, apply to all editions of the object.

This support enables consistent audit policy definition and enforcement in all object editions, whether they are existing or new. Applications which rely on edition-based redefinition can take advantage of the availability of the audit policies to audit all editioned objects in a consistent manner.

Related Topics

- [Oracle® Database Security Guide](#)

