Perspectives on Oracle Database Scalability
by John Kanagaraj

New Security Features in Oracle Database 12c Release 1
by Jim Czuprynski

Evaluation of Virtualization
by Francisco Munoz Alvarez
Tighter Than Ever: New Security Features in Oracle Database 12c Release 1

By Jim Czuprynski
Arup Nanda, Editor

Oracle Database 12c Release 1 (12cR1) is shaping up to be a major sea change for Oracle DBAs because it offers so many new features, especially when it comes to managing the security of Oracle databases. Some of the more glaring issues I’ve noticed in prior releases that needed to be addressed include:

Excessively Broad SYSDBA Privileges. The SYSDBA privilege is essentially a blank check for any Oracle user account to which it has been granted to perform just about any operation. However, that same SYSDBA privilege is currently necessary for quite a few administrative tasks in Oracle 11gR2, including the establishment and management of Oracle Data Guard physical and logical standby database environments; performing backup and restore actions via Oracle Recovery Manager (RMAN); and even for managing Transparent Data Encryption (TDE) wallets and decryption features.

Database Auditing: One Basic Need, Multiple Solutions. Thankfully, Oracle 11gR1 activated standard database auditing by default whenever a new database was created, while Oracle 11gR2 added some excellent security features, including automatic auditing of specific system privileges and activities, as well as monitoring for modification of existing and valid auditing policies. But the dizzying array of different auditing options, especially Fine-Grained Auditing (FGA) policies enforced via package DBMS_FGA, sometimes makes auditing for suspicious activity more convoluted and confusing than necessary. What disconcerts me most is that an unscrupulous DBA could attempt to cover his or her tracks by simply deleting suspicious entries from the internal database tables (SYS.AUD$ and SYS.FGA_LOG$) that comprise the audit trails.

Fortunately, Oracle 12cR1 offers specific solutions to these issues, and, in many cases, the solutions are relatively painless to implement. I’ll start with an exploration of several new system privileges that limit SYSDBA privileges to all but a select few.

Tightened Privileges: SYSBACKUP, SYSDG and SYSKM

Oracle 12cR1 adds three new sets of system privileges that tightly restrict the necessary rights to perform RMAN operations, manage Data Guard environments and handle TDE tasks.

SYSBACKUP: Just for RMAN. To perform any RMAN backup, restoration or recovery operations, Oracle 11gR2 required the user executing these activities to have been granted the SYSDBA privilege. Oracle 12cR1 now provides the SYSBACKUP privilege with a limited scope extending only to backup and recovery operations.

SYSDG: Data Guard Only. The SYSDBA privilege is also required to set up and manage a Data Guard environment in Oracle 11gR2. To provide a clearer separation of duties, Oracle 12cR1 now provides the SYSDG privilege that’s designed just for management of Data Guard environments.

SYSKM: Well Overdue. In Oracle 11gR2, the SYSDBA privilege is still needed to permit a user to open the Oracle “wallet” for TDE to allow access to encrypted data within the database. To provide a clear separation of duties, Oracle 12cR1 now limits TDE management tasks to only the SYSKM privilege.

Table 1 below summarizes the privileges that these three new privilege groups provide.

<table>
<thead>
<tr>
<th>Privilege</th>
<th>SYSBACKUP</th>
<th>SYSDG</th>
<th>SYSKM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database STARTUP and SHUTDOWN</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>RMAN BACKUP and RESTORE operations</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>RECOVER Operations</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>ALTER DATABASE ARCHIVELOG</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Perform FLASHBACK DATABASE and remove Guaranteed Restore Points</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>View DBA_* , GV$* and V$* views (but not schema-owned data)</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>CREATE or DROP DATABASE</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Create a Control File</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Create a PFILE / SPFILE</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Maintain SYSAUX tablespace during database upgrade operations</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Audit any operation</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Start the Fast-Start Failover Observer and invoke procedure INITIATE_FS_FAILOVER for DBMS_DRS package to force a failover</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Execute Data Guard Broker Manager (DGMBR) commands</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Manage Primary and Standby database instances</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Make trusted callouts to LogMiner packaged procedures and functions</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Manage any and all TDE operations</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>View TDE information</td>
<td>☑</td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

Table 1: SYSBACKUP, SYSDG and SYSKM Privileges

continued on page 18
Password File Support

These new administrative privileges are also supported during the creation or upgrade of a database’s password file via the Oracle 12cR1 version of the orapwd command. For example, this code would convert an existing Oracle 11gR2 password file for the orcl database to its Oracle 12cR1 equivalent, setting up a SYSBACKUP and SYSDG entry in the new password file:

```bash
$ orapwd -format 12 -sysbackup y -sysdg y -inputfile $ORACLE_HOME/dbs/pwdorcl
```

Analyzing and Capturing Errant Privileges

While the new SYSBACKUP, SYSDG and SYSKM privileges certainly help to segregate selected activities appropriately, Oracle DBAs also need to worry about whether privileges have been granted too liberally to user accounts in the past, regardless if these privileges have ever been used. Oracle 12cR1 tackles this challenge via the new DBMS_PRIVILEGE_CAPTURE package and its related feature set:

- The CREATE_CAPTURE procedure constructs a privilege analysis policy that will monitor either all privileges used within the entire database, just those used via a specific security role or even those used based on specific application contexts.
- A privilege capture policy is enabled via procedure ENABLE_CAPTURE. Once either sufficient time has passed or application activity has completed, procedure DISABLE_CAPTURE halts the analysis.
- Procedure GENERATE_RESULT flushes all privilege analysis findings to the database's data dictionary, and then the analysis results can be reviewed by querying the data dictionary views listed in Table 2. Be sure to note the neatest ability of this new privilege analysis feature set: the capability to review system and object privileges that have been previously granted to users but have never been used.
- Finally, after privilege capture is completed and the policy has been disabled, the DROP_CAPTURE procedure drops the analysis policy as well as any used or unused privilege data already captured.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA_PRIV_CAPTURES</td>
<td>Lists all of the database’s privilege analysis policies</td>
</tr>
<tr>
<td>DBA_USED_PRIVS</td>
<td>Lists all privileges that were captured as used during last run of DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT</td>
</tr>
<tr>
<td>DBA_USED_SYS_PRIVS</td>
<td>Contains all system privileges that were captured as used during last run of DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT</td>
</tr>
<tr>
<td>DBA_USED_OBJECT_PRIVS</td>
<td>Contains all object privileges that were captured as used during last run of DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT</td>
</tr>
<tr>
<td>DBA_USED_SYS_PRIVS_PATH</td>
<td>Contains all system privileges that were used for privilege analysis policies during the last run of DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT</td>
</tr>
<tr>
<td>DBA_USED_OBJECT_PRIVS_PATH</td>
<td>Contains all object privileges that were used for privilege analysis policies during the last run of DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT</td>
</tr>
<tr>
<td>DBA_UNUSED_SYS_PRIVS</td>
<td>Contains all system privileges that have been GRANTed but have never been used</td>
</tr>
<tr>
<td>DBA_UNUSED_OBJECT_PRIVS</td>
<td>Contains all object privileges that have been GRANTed but have never been used</td>
</tr>
</tbody>
</table>

Table 2: Privilege Analysis Reporting Views

Database Auditing: The Great Simplification

I’ve long been a strong proponent of using Fine-Grained Auditing (FGA) features to audit for inappropriate user access to data. However, implementing FGA and even standard auditing features is decidedly non-trivial even in Oracle 11gR2. Thus, I was pleasantly surprised to see that Oracle 12cR1 has completely revamped the methodology for capturing and retaining audit trail information to insure that proper levels of auditing are always in place and that auditing standards can be maintained with minimum effort.

Capturing Audit Information

The first major change to auditing is the way that Oracle 12cR1 collects auditing information. Whenever an auditable event occurs, one of two named queues within each 12cR1 database’s SGA collects the data; whenever one of the queues overflows, its data is flushed to persistent storage. These data are stored in tables owned by the new audit schema owner (AUDSYS), which also owns the named queues.

The second queue is therefore free to continue to collect audit information while the first queue’s content is written to disk. While this queued write mode — the default method — makes for extremely efficient collection of audit data, it’s still possible to lose critical audit data during an instance crash, so Oracle 12cR1 also provides an immediate-write mode as well for systems that need to preserve audit information at the cost of lower performance. Note that it’s also possible to flush the audit trail contents in queued write mode immediately to their persistent tables via procedure FLUSH_UNIFIED_AUDIT_TRAIL of the new DBMS_AUDIT_MGMT package.

Audit Policies

Oracle 12cR1’s new audit policy completely upgrades the methods by which database objects, privileges and statements are chosen (or excluded!) as targets for auditing. The new CREATE AUDIT POLICY command is used to construct a policy object that audits for usage of specific system privileges, privileges granted via a specific role, specific RDBMS actions (e.g., CREATE TABLE, ALTER INDEX), or any combination of these. For example:

```sql
-- Build audit policies for:
-- 1.) Privileges accessed
-- 2.) Actions performed
-- 3.) Roles accessed
-- 4.) A combination of all three

CREATE AUDIT POLICY aud_ltd Sys_priv
PRIVILEGES
CREATE ANY TABLE,
CREATE ANY INDEX,
ALTER ANY TABLE,
ALTER ANY INDEX;

CREATE AUDIT POLICY aud_ltd_sys_acts
ACTIONS
DISABLE TRIGGER,
ALTER SYSTEM;

CREATE AUDIT POLICY aud_ap_roles
ROLES
ap_maint,
ap_read_only;

CREATE AUDIT POLICY aud_mixed_privactroles
PRIVILEGES
DROP ANY TABLE, DROP ANY INDEX
ACTIONS
ENABLE TRIGGER, ALTER SYSTEM
ROLES
ap_maint, ap_read_only;
```
Of course, it’s also possible to audit one or more database objects for specific usage as well. Here’s an example of an audit policy that monitors any updates and deletes a single AP schema object, APINVOICES. Note that it also applies specific conditions under which auditing will occur — in this case, on a per-statement basis, but only when the MODULE attribute of a user’s session is set to APBATCH:

```
CREATE AUDIT POLICY aud_ap_invoices
ACTIONS UPDATE, DELETE ON ap.invoices
WHEN 'SYS_CONTEXT("USERENV", "MODULE")' = 'APBATCH'
EVALUATE PER SESSION;
```

Just like the Fine-Grained Audit policies created via DBMS_FGA in prior releases, an audit policy can also call a handler procedure that performs additional notifications or logging. It’s also possible to whitelist (i.e., exclude) specific users from an audit policy, thus making the creation of separate, less restrictive audit policies for just a few users completely unnecessary.

The original AUDIT and NOAUDIT commands have been re-tasked to respectively enable and disable existing 12cR1 audit policies. The DROP AUDIT POLICY command is used to drop an existing audit policy, but only after it’s been disabled via the NOAUDIT command. Best of all, implementing Oracle 12cR1 audit policies initially is less worrisome because there are three predefined audit policies that will suffice for most auditing requirements. I’ve summarized them in Table 3 below.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORA_SECURECONFIG</td>
<td>Audits for all standard system privileges and RDBMS actions,</td>
</tr>
<tr>
<td></td>
<td>including modification of existing audit policies, enabled by default</td>
</tr>
<tr>
<td>ORA_ACCOUNT_MGMT</td>
<td>Audits for creation of and changes to user accounts and roles</td>
</tr>
<tr>
<td>ORA_DATABASE_PARAMETER</td>
<td>Audits for SPFILE creation, ALTER DATABASE and ALTER SYSTEM commands</td>
</tr>
</tbody>
</table>

Table 3: Predefined Audit Policies

Finally, the AUDIT_UNIFIED_POLICIES view records information about all audit policies, including whether they’re active, which users will be audited, and whether an audit action is based on the success or failure of the audited operation.

Unified Audit Trails: BAI and EAI

Oracle 12cR1 combines basic audit information (BAI) — essentially the same database session and operation level auditing that was available in earlier releases — with a new set of previously under-audited or unaudited operations. The results of Fine-Grained Auditing (FGA) as well as audited information about Data Pump Exports and Imports, RMAN BACKUP and RESTORE operations, Oracle Label Security (OLS) activity, and operations related to Real Application Security (RAS) and Database Vault (DV) are now identified as extended audit information (EAI). EAI results are combined into a single, unified audit trail view — appropriately named UNIFIED_AUDIT_TRAIL — and owned by the new AUDSYS user account. In addition, the underlying table for UNIFIED_AUDIT_TRAIL may only be opened in READ ONLY mode, it can’t be modified by any other user than AUDSYS, and can only be updated during database upgrades. This prohibits deliberate or accidental tampering with the audit trail records.

Whether an Oracle 12cR1 database has been upgraded from an earlier version or created brand-new, it’s operating in mixed auditing mode by default, which means that it’s still using traditional pre-12cR1 auditing features. The only difference is that the new database uses the new ORA_SECURECONFIG audit policy to handle audit operations. While this is certainly acceptable, there are some additional mandatory tasks required to activate unified auditing mode:

- First, shut down all Oracle Database processes and instances.
- Relink the Oracle binaries in the selected Oracle 12cR1 database home with the uniaud_on option.
- If desired, create a new tablespace for UNIFIED_AUDIT_TRAIL (by default, its base tables are written into the SYSAUX tablespace).

Note that once unified audit mode is fully activated, the pre-12cR1 initialization parameters that control auditing (e.g. AUDIT_TRAIL, AUDIT_SYS_OPERATIONS) won’t be honored, as they are essentially meaningless, because audit policies now control the scope of what’s actually being audited.

Prior releases required the SYSDBA privilege for configuring and controlling auditing and viewing the audit trails. Oracle 12cR1 now provides two new security-related roles: AUDIT_ADMIN, which allows the granted user the ability to configure audit settings as well as manage the audit trail via the DBMS_AUDIT_MGMT package; and AUDIT_VIEWER, which permits just the ability to view and analyze the unified audit trail.

Audit Trail Maintenance

Unlike earlier releases, in Oracle 12cR1 all audit trail maintenance is performed via procedures in package DBMS_AUDIT_MGMT:

- Procedure CREATE_PURGE_JOB creates an automated task to purge audit trail records for either a 12cR1 container database or a pluggable database. It can also restrict purging based on the timestamp of each audit trail entry to limit inadvertent deletions.

- Procedure CLEAN_AUDIT_TRAIL performs a manual purge of the unified audit trail.

What’s Next for 12cR1?

In next issue’s installment, I’ll turn my attention to some of the most significant upgrades to the Oracle optimizer in Oracle 12cR1, including Automatic Re-Optimization (ARO), Adaptive Execution Plans (AEP) and SQL Plan Directives (SPD).

---

**About the Author**

Jim Czuprynski has accumulated more than 30 years of experience during his career in information technology. He served diverse roles at several Fortune 1000 companies in those three decades — mainframe programmer, applications developer, business analyst and project manager — before becoming an Oracle Database administrator in 2001. He currently holds OCP certification for Oracle 9i, 10g and 11g. Jim teaches the core Oracle University database administration courses on behalf of Oracle and its education partners throughout the United States and Canada. He was selected as Oracle Education Partner Instructor of the Year in 2009. He continues to write a steady stream of articles that focus on myriad facets of Oracle Database administration, with nearly 100 articles to his credit since 2003 at databasejournal.com. Jim’s monthly blog, Generally … It Depends (http://jimczuprynski.wordpress.com), contains his regular observations on all things Oracle. Jim has presented topics at Oracle OpenWorld (2008 and 2013), IOUG’s COLLABORATE conferences (2011 and 2013) and OUG Norway (2013).