MUNIQSOFT

Best Practices für das Datenbank-Audit in Oracle 11g und 12c

Dr. Elke Fritsch

Muniqsoft GmbH

- **Tätigkeitsbereiche:**
  - **Oracle IT-Consulting & Services**
  - **Oracle Remote Support und Rufbereitschaft**
  - **Oracle Schulungen (SQL, PL/SQL, DBA, APEX, … - gerne auch Inhouse)**
  - **Oracle Lizenzen**

Muniqsoft GmbH
Schulungszentrum
Grünwalder Weg 13a
82008 Unterhaching
Tel.: 089 / 679090 40

MUNIQSOFT

Muniqsoft GmbH
IT-Consulting & Support
Witneystr. 1
82008 Unterhaching
Tel.: 089 / 6228 6789 0

Um was geht's hier ?

- **Das hier vorgestellte fiktive Audit-Szenario einer fiktiven Firma soll dazu dienen**
  - **die Probleme mit den Default-Einstellungen und der Konfiguration des Audits in den Versionen 11g und 12c aufzuzeigen**
  - **ein paar Tipps zur Konfiguration zu geben, die**
    - **ein zielgerichtetes und möglichst vollständiges Auditing ermöglichen**
    - **die Auswertung einfacher machen**
    - **das Housekeeping so gestalten, dass einem die Audit-Daten nicht über den Kopf wachsen**
  - **Für die Tests wurde die Standard Edition verwendet (für 12c die SE2 mit dem neuesten PSU-Patch ohne Multitenant).**

Es wirken mit

IT-Consultant

Geschäftsführer

Systemadministrator
und Teilzeit-DBA

diverse Mitarbeiter mit Zugriff
auf die Datenbank

der neue DBA

ein Werkstudent

Status Quo

- Im Einsatz sind Oracle-Datenbanken der Version 11.2.0.4 (Standard Edition One) auf Linux und auf Windows.
- Alle Angestellten der Firma loggen sich unter dem selben Account (Kunden) ein und greifen über den SQL-Developer auf die Tabellen zu.
- Der Enterprise Manager ist konfiguriert, wird aber selten benutzt.
- Die beiden DBAs und der Consultant arbeiten als SYSDBA auf der Instanz, alle anderen, auch der Werkstudent, haben die Rollen CONNECT und RESOURCE
  - Die User legen gerne Indizes an, erstellen haufenweise Zwischentabellen und basteln z.T. auch eigene Prozeduren und Trigger.
- Audit ist zur Zeit aktiviert, wird aber nicht ausgewertet.
- Die Audit-Daten werden über einen Job jeden Monat gelöscht.
- Alle Audit-Parameter stehen auf Default-Werten.

Was soll auditiert werden ?

- Stops und Starts der Instanz
- Änderungen an Parametern und Datenbankdateien
- fehlgeschlagene Login-Versuche
- Rechte-Vergabe bzw. Entzug
- Inserts, Updates und Deletes auf der Tabelle Kundendaten inclusive des SQL-Befehls
- Änderungen an Tabellen und PL/SQL-Code in den Schemata KUNDEN und KD_VERW (Kundenverwaltung)
- Für einen begrenzten Zeitraum alle Aktionen des Werkstudenten, der eine Applikation für die Kundenverwaltung entwickeln soll und hauptsächlich nachts werkelt.
- Alle Aktionen des Consultants.

Was deckt der Default in 11g ab ?

Aktion	**wird auditiert**
fehlgeschlagene Login-Versuche	✔
DML-Aktionen an User-Tabellen	✘
DDL-Aktionen an User-Tabellen	nur mit ANY-Rechten
Änderungen am PL/SQL-Code von Usern	nur mit ANY-Rechten
Stop und Start der Datenbank	✔
Aktionen des Users SYS (außer Start, Stop)	✘
Änderungen an Parametern und Datenbankdateien	✔
Rechte-Vergabe bzw. Entzug	nur mit ANY-Rechten

Audit von Stops und Starts der Instanz

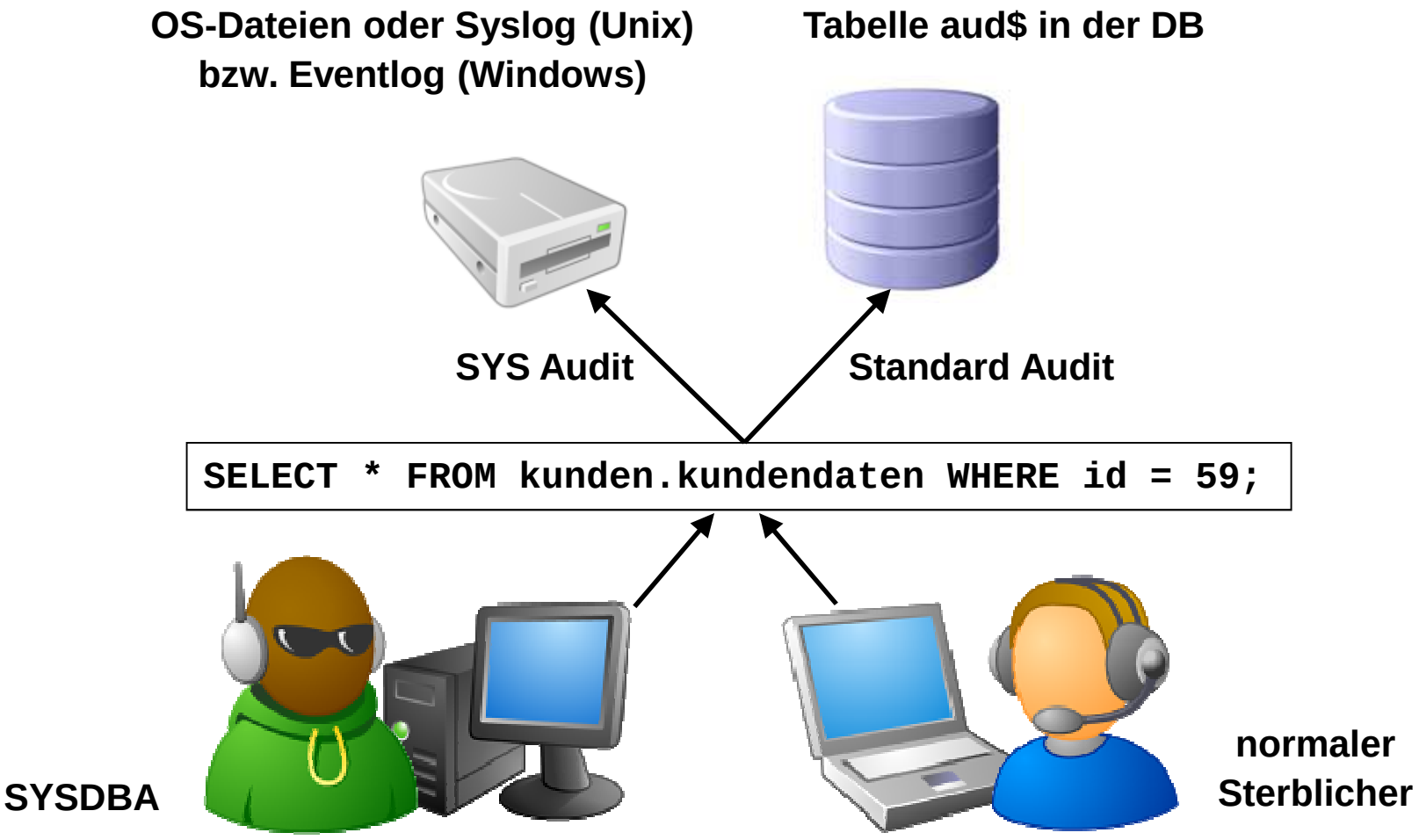
- **Gehört zum Mandatory Auditing**
  - **Automatisches Auditing, unabhängig von den audit_trail-Einstellungen, kann nicht vom DBA oder anderen unterdrückt werden**
  - **zeichnet Anmeldungen von Usern mit SYSDBA- oder SYSOPER-Rechten, Instanz-Starts und -stops auf.**
  - **Unter Unix werden die Informationen per Default in das durch `AUDIT_FILE_DEST` angegebene Verzeichnis geschrieben, in Dateien mit Namen `ora_<pid>.aud` (pid: Prozessnummer des Serverprozesses).**
  - **Alternativ kann man auch das Syslog verwenden**
  - **Unter Windows erscheinen die Einträge im Eventlog.**
- **Problem**
  - **Weder unter Linux noch unter Windows sind die erzeugten Files besonders leicht auszuwerten.**

SYS Auditing

- **Seit Oracle 9i können Aktivitäten von Benutzern mit SYSDBA- und SYSOPER-Privileg überwacht werden.**
  - **Dazu muss der statische Initialisierungsparameter `AUDIT_SYS_OPERATIONS` auf TRUE gesetzt werden.**
  - **Das Sys-Auditing funktioniert auch, wenn audit_trail auf NONE steht.**
  - **Gespeichert werden die Einträge wie beim Mandatory Auditing.**
  - **Es werden immer alle Statements vollständig protokolliert (auch Selects).**

- **Probleme:**
  - **mühsame Auswertung wie beim Mandatory Auditing**
  - **Das Audit ufert wegen der vielen recursive calls schnell aus.**
    Z.B. erzeugt der RMAN tonnenweise Audit-Einträge, wenn er sich in Skripten als sysdba mit der Datenbank verbindet.

Wie bekommt man das unter einen Hut ?



"Unified Auditing" unter 11g

- Umstellung des Audits auf "XML" bzw. "XML, extended"
- Vorteile:
  - Standard Audit, Mandatory Auditing, Sys Auditing und Fine Grained Auditing können über die View DBA_COMMON_AUDIT_TRAIL gemeinsam ausgewertet werden.
  - Oracle wertet die Audit-XML-Dateien genauso aus wie das Alert- und Listener.log im XML-Format unter 11g → kein Parsen nötig
  - XML-Audit ist performanter als tabellenbasiertes Audit.
  - Housekeeping wird vereinfacht, weil nur noch an einem Platz aufgeräumt werden muss.

"Unified Auditing" unter 11g einrichten

- **Beispiel unter Windows**
  - `ALTER SYSTEM SET audit_trail = xml, extended SCOPE = SPFILE;`
  - `ALTER SYSTEM SET audit_file_dest = 'c:\temp' DEFERRED;`
  - `ALTER SYSTEM SET audit_sys_operations = true SCOPE = SPFILE;`
  - `SHUTDOWN IMMEDIATE`
  - `STARTUP`

- **Um den Audit-Trail vor den potentiell gefährlichen DBAs zu schützen, sollte man zumindest dafür sorgen,**
  - **dass diese personalisierte OS-Accounts haben und /oder**
  - **dass die XML-Files zeitnah ausgewertet und danach auf einen anderen Server transferiert werden**

Audit-Admin-User oder Rolle einrichten

- Optional, aber sehr empfehlenswert
- Beispiel (mit garantiert unverdächtigem Namen):

```
CREATE USER udita IDENTIFIED BY <pwd>;
GRANT CREATE SESSION, AUDIT SYSTEM, CREATE JOB,
      SELECT_CATALOG_ROLE,CREATE SYNONYM,
      CREATE VIEW, CREATE TABLE, CREATE PROCEDURE TO udita;
GRANT SELECT ON v_$xml_audit_trail TO udita;
GRANT SELECT ON dba_common_audit_trail TO udita;
GRANT SELECT ON dba_audit_mgmt_last_arch_ts TO udita;
GRANT EXECUTE ON DBMS_AUDIT_MGMT TO udita;
```

- Denken Sie daran, dass jeder Select des SYS-Users aufgezeichnet wird und den Audit-Trail noch mehr aufbläst.
- Werten Sie ihn deshalb immer als Audit-User oder als SYSTEM aus !

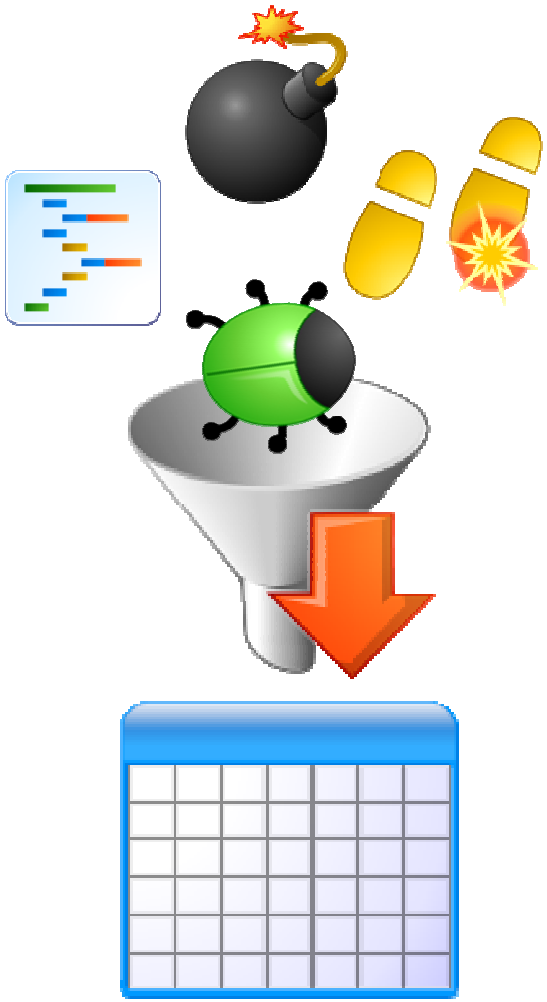
Housekeeping einrichten

```
BEGIN  /* Job, der den Timestamp setzt, bis zu dem Daten gelöscht werden */
  DBMS_SCHEDULER.CREATE_JOB (
                 job_name => 'set_last_archive_timestamp',
                 job_type => 'PLSQL_BLOCK',
               job_action => 'BEGIN '||
                                 'DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP('||
                                 'DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML,'||
                                 'TRUNC(SYSDATE)-31); END;',
               start_date => sysdate,
          repeat_interval => 'freq=daily;byhour= 0;',
                  enabled => TRUE);
END;
BEGIN    /* eigentlicher Purgejob */
   DBMS_AUDIT_MGMT.CREATE_PURGE_JOB(
                audit_trail_type => DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML,
      audit_trail_purge_interval => 24 /* Stunden */,
          audit_trail_purge_name => 'daily_purge_job',
         use_last_arch_timestamp => TRUE);
END;
```

DBA_COMMON_AUDIT_TRAIL

- **Diese View umfasst Einträge aus**
  - **DBA_AUDIT_TRAIL (Quelle: sys.aud$)**
  - **DBA_FGA_AUDIT_TRAIL (Quelle: sys.fga_log$)**
  - **V$XML_AUDIT_TRAIL (Quelle: Files im audit_file_dest)**

- **Bei eingeschaltetem SYS-Auditing besteht das größte Problem darin, neben den Unmengen von SELECTs, die der SYSDBA absetzt, die interessanten herauszufinden.**



Beispiel Audit-Trail-Übersicht

- **Lösungsvorschlag**
  - **View für alle außer den "unverdächtigen" Selects (recursive calls, RMAN-Selects etc.) erstellen.**
- **Vorarbeit**
  - **Als Audit-User das Audit einige Tage laufen lassen (mit allen Backup-Skripten etc.) und dann eine Tabelle mit den SQL-Befehlen anlegen.**

```
CREATE TABLE sqls AS
SELECT DISTINCT SUBSTR(TRIM(sql_text), 1,400) sql_text
FROM v$xml_audit_trail
WHERE sql_text IS NOT NULL ORDER BY 1;

ALTER TABLE sqls ADD CONSTRAINT sql_pk PRIMARY KEY(sql_text);
```

  - **Danach die Tabelle bearbeiten und alles außer recursive SQL und Package-Aufrufen etc. entfernen.**

Beispiel Audit-Trail-Übersicht (f)

- **Als Audit-User erstellen:**

```
CREATE OR REPLACE VIEW trail AS
SELECT TO_CHAR(extended_timestamp, 'dd.mm.rr hh24:mi:ss') datum,
       db_user, os_user, userhost,
       REGEXP_SUBSTR(comment_text, '[^=)]*',1,9) ip_address,
       REGEXP_SUBSTR(comment_text, '[^=)]*',1,13) port,
       statement_type, session_id, returncode, ses_actions, scn,
       object_schema, object_name, new_name, priv_used,
       obj_privilege, sys_privilege, grantee,
       audit_option, SUBSTR(TRIM(sql_text), 1,400) sql_text
FROM dba_common_audit_trail t
-- Ausschluß uninteressanter Selects
WHERE NOT EXISTS (SELECT 1 FROM sqls s
                  WHERE s.sql_text = t.sql_text)
AND action NOT IN (101,102) -- ohne LOGOFF und LOGOFF BY CLEANUP
ORDER BY extended_timestamp DESC;
```

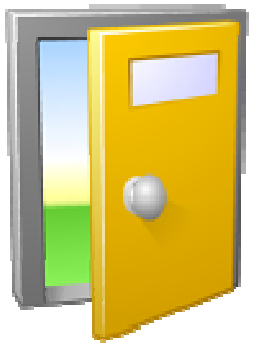
Audit von DBA-Aktionen

- **Stops und Starts der Instanz sowie Änderungen an Parametern und Datenbankdateien werden per Default auditiert.**

```
CREATE OR REPLACE VIEW trail_db_change AS
SELECT datum, db_user, os_user, userhost, ip_address, port,
       statement_type, returncode, priv_used, os_privilege,
       sql_text
FROM trail
-- beliebig erweiterbare Liste
WHERE REGEXP_LIKE(UPPER(sql_text),
     'ALTER SYSTEM|CREATE PFILE|CREATE SPFILE|'||
     'CREATE DISKGROUP|ALTER DISKGROUP|DROP DISKGROUP|'||
     'CREATE DATABASE|ALTER DATABASE|DROP DATABASE|'||
     'ALTER TABLESPACE|CREATE TABLESPACE|DROP TABLESPACE|'||
     'CONNECT|STARTUP|SHUTDOWN');
```

Session-Audit anpassen

- Per default werden alle LOGON, LOGON bzw. LOGOFF BY CLEANUP-Vorgänge aller User außer sys aufgezeichnet.
- 2 User erzeugen fleißig Session-Audit-Einträge, auch wenn nichts auf der Instanz läuft.
  - Die EM-User DBSNMP und SYSMAN
- Man kann diese User entweder ganz vom Session-Audit ausschließen
  - `NOAUDIT CREATE SESSION BY dbsnmp, sysman;`
- oder nur erfolglose Anmeldeversuche dieser User auditieren
  - `NOAUDIT CREATE SESSION BY dbsnmp, sysman;`
  - `AUDIT CREATE SESSION BY dbsnmp, sysman WHENEVER NOT SUCCESSFUL;`



Auswertung des Session-Audits

- **Fehlgeschlagene Logins**

```
SELECT COUNT(*), os_user, db_user, userhost, returncode,
       SUBSTR(datum,1,8)
FROM trail
WHERE statement_type = 'LOGON' AND returncode <> 0
GROUP BY os_user, db_user, userhost, returncode,
        SUBSTR(datum,1,8);
```

- **Anmeldeversuche mit nicht existierenden Benutzern**

```
SELECT os_user, db_user, userhost, returncode, datum
FROM trail t
WHERE statement_type = 'LOGON'
AND NOT EXISTS (SELECT 1 FROM dba_users u
                WHERE t.db_user = u.username);
```

Audit der Rechte- und Userverwaltung

- **Was fehlt bei den Defaults ?**
  - **Audit für Änderungen des eigenen Passworts**
  - **Audit für Aktionen wie Grants auf Directories, selbst erstellte Prozeduren etc.**

- **Tipps für die Erweiterung des Audits**
  - `AUDIT GRANT DIRECTORY, GRANT PROCEDURE, GRANT SEQUENCE, GRANT TABLE, GRANT TYPE;`
  - **Damit wird das Erstellen, Ändern und Löschen von Rollen (im Default enthalten) sowie die Vergabe und der Entzug von Rechten an Directories, Prozeduren, Funktionen, Packages, Sequenzen, Tabellen, Views, Materialized Views und Typen auditiert.**

Audit der Rechte- und Userverwaltung (f)

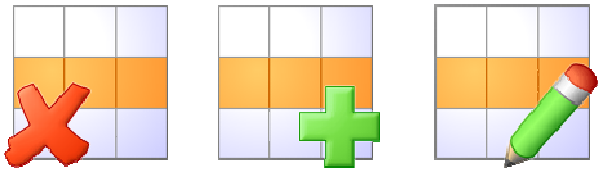
- **Beispiel für die Auswertung:**

```
CREATE OR REPLACE VIEW trail_user_admin
AS
SELECT datum, db_user, os_user, userhost, ip_address,
       port,statement_type,returncode, object_name,
       priv_used, os_privilege,sys_privilege,
       grantee, sql_text
FROM trail
WHERE REGEXP_LIKE(UPPER(sql_text),
 'GRANT|REVOKE|ROLE|CREATE USER|ALTER USER|DROP USER');
```

DML-Aktionen auditieren

- **DML-Audit für Zugriffe auf die Tabelle KUNDENDATEN**
  - `AUDIT INSERT, UPDATE, DELETE ON kunden.kundendaten;`
- **Auswertungsbeispiel:**
  - **Wenn man nur die Spalte Object_name auswertet, gehen einem die Aktionen des Users SYS durch die Lappen.**
  - **Auswertung mit Auflösung der ses_actions-Spalte s. Notizteil.**

```
CREATE OR REPLACE VIEW trail_dml_kundendaten AS
SELECT os_user, db_user, userhost, datum,
       ses_actions, session_id, returncode,
       scn, sql_text
FROM trail
WHERE UPPER(sql_text) LIKE '%KUNDENDATEN%'
ORDER BY datum DESC;
```



DML-Aktionen auditieren (f)

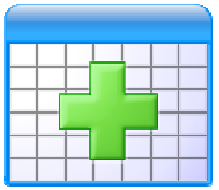
- **Verbesserte Auswertung mit Auflösung des ses_actions-Eintrags**

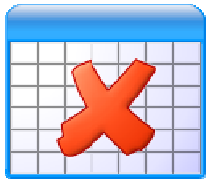
```
SELECT os_user,db_user,userhost,
       TO_CHAR(extended_timestamp,'dd.mm.rr hh24:mi:ss') datum,
       CASE REGEXP_INSTR(ses_actions, '[^-]+')
                    WHEN  4 THEN 'DELETE'
                    WHEN  7 THEN 'INSERT'
                    WHEN 11 THEN 'UPDATE'
       END||' '||
       CASE REPLACE(ses_actions, '-')
            WHEN 'S'
            THEN 'successful'
            ELSE 'failed'
       END aktion, session_id, returncode, scn, sql_text
FROM dba_common_audit_trail
WHERE object_name = 'KUNDENDATEN'
ORDER BY datum DESC;
```

DDL-Änderungen auditieren

- Bringt dieser Befehl das gewünschte Ergebnis ?
  - `AUDIT TABLE, PROCEDURE BY kunden;`

- Nicht wirklich !
  - `TABLE` steht für `CREATE`, `DROP` und `TRUNCATE TABLE`, aber nicht `ALTER`, z.B. `ALTER TABLE ...RENAME`
  - `PROCEDURE` steht für `CREATE` bzw. `DROP FUNCTION | PROCEDURE | PACKAGE | PACKAGE BODY | LIBRARY`
  - Das Erstellen, Löschen und Ändern von Triggern, Views und Indizes wird hiermit nicht auditiert
  - DDL-Aktionen anderer User in den beiden Schemata werden dagegen abgedeckt, weil dafür ANY-Rechte erforderlich sind.





Trigger

DDL-Änderungen auditieren (f)

- **Nachbesserung**

```
NOAUDIT TABLE BY kunden;
AUDIT INDEX, VIEW, TABLE, TRIGGER, ALTER TABLE BY kunden;
```

- **Auswertungsbeispiel:**

```
SELECT os_user, db_user, userhost, datum,
       object_name, priv_used, session_id, returncode,
       scn, sql_text
FROM trail
WHERE object_schema IN ('KUNDEN', 'KD_VERW')
ORDER BY datum DESC;
```

  - **Der Returncode ist übrigens auch dann 0, wenn der PL/SQL-Code erfolglos kompiliert wurde.**

Audit "normaler" User

- Der DBA wüsste gerne, was der Werkstudent so treibt.
- Er probiert's mal mit
  - `AUDIT ALL BY kd_verw;`
- Was ist jetzt alles aktiviert ?

```
SELECT audit_option
FROM dba_common_audit_trail
WHERE action = 104 -- Nummer für die Audit-Aktivierung
ORDER BY audit_option;
```

- Problem:
  - Der Werkstudent hat die Rollen RESOURCE und CONNECT und darüber hinaus SELECT-und DML-Rechte im Schema Kunden, deshalb nützt dieses Audit herzlich wenig.

Audit "normaler" User (2)

- Sinnvoller ist es, die aufzuzeichnenden Aktionen direkt anzugeben, z.B.
  - `NOAUDIT ALL BY kd_verw;`
  -
```
AUDIT TABLE, ALTER TABLE, UPDATE TABLE, INSERT TABLE,
      DELETE TABLE, TRIGGER, TYPE, PROCEDURE,
      EXECUTE PROCEDURE
BY kd_verw BY ACCESS;
```

- Kontrolle mit

```
SELECT audit_option FROM dba_stmt_audit_opts
WHERE user_name = 'KD_VERW';
```

Was noch zu beachten ist

- **Änderungen von Audit-Einstellungen haben häufig nicht die gewünschte Wirkung**
  - **Dieser Befehl aktiviert das Audit von DML-Aktionen auf allen Tabellen**
  - `AUDIT INSERT TABLE, UPDATE TABLE, DELETE TABLE BY ACCESS;`
  - **Wenn man nachträglich die Aktionen des Users Kunden an seinen eigenen Tabellen ausschließen will, so bringt der folgende Befehl nichts**
  - `NOAUDIT INSERT TABLE, UPDATE TABLE, DELETE TABLE BY kunden;`
- **Wichtig:**
  - **Bei Änderungen immer die Originalanweisung rückgängig machen und dann das Audit neu einstellen.**
  - **Die Noaudit-Option macht nur Audit-Einstellungen mit derselben Syntax rückgängig und greift nur für neue Sessions.**

Alles neu mit 12c ?

- Nach der Migration auf 12c gelten die alten Einstellungen weiter.
- Zusätzlich sind in diesem sog. *Mixed Mode* allerdings 2 Policies des neuen Unified Auditing (UA) aktiv.
  - `ORA_SECURECONFIG` (entspricht den erweiterten Defaults aus 11g)
  - `ORA_LOGON_FAILURES`
- Probleme mit dem Mixed Mode
  - Audit-Daten werden in unterschiedlichen Zielen gesammelt.
  - → die Auswertung wird mühsamer
  - → das Housekeeping wird aufwendiger
- Wenn man beim alten Modus bleiben will, sollte man die UA-Policies deaktivieren, um den zusätzlichen Overhead zu vermeiden.
  - `NOAUDIT POLICY ora_secureconfig;`
  - `NOAUDIT POLICY ora_logon_failures;`

Was hat sich (u.a.) geändert ?

- Der Audit-Trail gehört dem gesperrten, read_only-Schema `AUDSYS`.
- Das Audit wird (wie Fine Grained Auditing) über Policies konfiguriert
- Die Benutzung des Syslogs oder Speicherung der Standard-Audit-Daten im Betriebssystem ist nicht mehr möglich.
- Die Daten werden zunächst in der SGA vorgehalten und erst nach Überschreiten einer bestimmten Größe (per Default 1 MB) auf Platte geschrieben.
- Wenn die Instanz nicht zum Schreiben geöffnet ist, werden binäre Audit Files ins Verzeichnis `$ORACLE_BASE/audit/$ORACLE_SID` geschrieben.
  - Beim Starten der Instanz werden sie in die Audit-Tabelle hochgeladen.
  - Die Prozedur `DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES` löscht sie auf OS-Ebene.

Gründe, den Status Quo beizubehalten

- **Man sollte (vorerst) bei der 11g-Methode bleiben, wenn**
  - **das Audit im Syslog aufgezeichnet werden soll.**
  - **die Auditdaten in OS-Dateien gespeichert und für die Auswertung und Speicherung an einen anderen Server weitergeleitet werden sollen.**
  - **man keine Zeit für gründliche Tests hat**
  - **man die Standard Edition One behalten will (oder muss). Diese ist nur noch für 12.1.0.1 erhältlich und die Aktivierung des Unified Auditing klappt hier nicht wirklich.**
  - **man auf die Standard Edition 2 umgestiegen ist.**
    - **Der Bug 18109788 (CLEANUP OF UNIFIED AUDIT TRAIL DOES NOT RELEASE LOB SEGMENT SPACE) läßt den SYSAUX-TS stark wachsen, wenn man nicht regelmäßig aufräumt.**
    - **Der Umzug der Audit-Strukturen auf einen anderen Tablespace, der hier Abhilfe schafft, funktioniert nur in der EE !!!**

Verschieben des Audit-Trails

- Geht nur in der Enterprise Edition (Stand Oktober 2016)

```
Oracle Database 12c Standard Edition Release 12.1.0.2.0 ...

CREATE TABLESPACE audit_ts
DATAFILE '/u01/oracle/oradata/o12c/audit_ts01.dbf'
SIZE 1G AUTOEXTEND ON NEXT 100M MAXSIZE 10g;

BEGIN
   DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION(
             audit_trail_type => dbms_audit_mgmt.audit_trail_unified,
   audit_trail_location_value => 'AUDIT_TS');
END;
/
→ ERROR at line 1:
ORA-46099: Feature wird nicht unterstützt oder ist nicht implementiert
```

Gründe für den Umstieg auf UA

- Der Audit-Trail kann nur über DBMS_AUDIT_MGMT gelöscht werden.
- Die View UNIFIED_AUDIT_TRAIL liefert (u.a.) zusätzlich Infos für Client-Identifier, das Client-Programm und verwendete DB-Links.
- Der User SYS wird per Default auditiert, aber wie ein normaler User !
  - → weniger Overhead durch recursive calls!!
  - Weil der RMAN sein eigenes Audit hat, kann man den Audit-Trail leichter auswerten bzw. filtern.
- Das SQL-Statement wird bei den auditierten Aktionen mit erfasst
- Rollen, Datapump und der SQL*Loader können auditiert werden.
- Die Syntax ist (etwas) übersichtlicher und flexibler.
- Schemaowner können das Audit nicht mehr selbst gestalten.
- Es gibt vordefinierte Rollen für die Verwaltung des Audits.

Was deckt der Default in 12c ab ?

Aktion	**wird auditiert**
fehlgeschlagene Login-Versuche	✔
DML-Aktionen an User-Tabellen	✘
DDL-Aktionen an User-Tabellen	ausgewählte Aktionen
Änderungen am PL/SQL-Code von Usern	ausgewählte Aktionen
Stop und Start der Datenbank	✔
Aktionen des Users SYS (außer Start, Stop)	ausgewählte Aktionen
Änderungen an Parametern und Datenbankdateien	✔
Rechte-Vergabe bzw. Entzug	nur mit ANY-Rechten

Erweitertes Mandatory Auditing

- **Auditiert werden unabhängig von der Audit-Konfiguration**
  - **alle Änderungen an der Audit-Konfiguration**
  - **CREATE /ALTER / DROP AUDIT POLICY sowie AUDIT und NOAUDIT**
  - **Die Benutzung des Packages DBMS_AUDIT_MGMT**
  - **Alle Änderungen an Database Vault-Einstellungen**
  - **Versuche, die Objekte des Audit-Trails zu manipulieren**
  - **Alle Connects, Starts und Stops der Instanz als SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM, bis die Datenbank zum Schreiben geöffnet ist.**
  - **(Fast) alle RMAN-Aktionen**

Audit-User einrichten

- **Durch die vorkonfigurierten Rollen wird das Erstellen des Audit-Users einfacher:**

```
DROP USER udita CASCADE;
GRANT CREATE SESSION, CREATE TABLE, CREATE JOB,CREATE VIEW,
      CREATE PROCEDURE, AUDIT_ADMIN, SELECT_CATALOG_ROLE
TO udita IDENTIFIED BY <pwd>;
GRANT SELECT ON unified_audit_trail TO udita;
GRANT EXECUTE ON DBMS_AUDIT_MGMT TO udita;
```

  - **Mit der AUDIT_ADMIN-Rolle kann man das Audit konfigurieren, überwachen und verwalten (Housekeeping etc.)**
  - **Nur die User sys und die Rollen audit_admin und admin_viewer haben Zugriff auf den unified_audit_trail.**
  - **Benutzer, die Audit Policies (auch für eigene Objekte) administrieren wollen, müssen das Recht AUDIT SYSTEM oder die Rolle AUDIT_ADMIN haben.**

Housekeeping einrichten

- Das Package DBMS_AUDIT_MGMT wurde in 12c verbessert, aber die in 11g eingerichteten Jobs funktionieren weiterhin
  - Man muss nur den Parameter audit_trail_type anpassen
  - Job, der den Timestamp setzt, bis zu dem Daten gelöscht werden

```
BEGIN
 DBMS_SCHEDULER.CREATE_JOB (
        job_name => 'set_last_archive_timestamp',
        job_type => 'PLSQL_BLOCK',
      job_action => 'BEGIN '||
                       'DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP('||
                       'DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,'||
                        'TRUNC(SYSDATE)-31); END;',
      start_date => sysdate,
 repeat_interval => 'freq=daily;byhour= 0;',
         enabled => TRUE);
END;
```

Housekeeping einrichten (f)

- **Timestamp erstmalig setzen**

```
BEGIN
  DBMS_SCHEDULER.RUN_JOB (
                job_name => 'set_last_archive_timestamp',
     use_current_session => FALSE);
END;
```

- **eigentlichen Purge-Job einrichten**

```
BEGIN
   DBMS_AUDIT_MGMT.CREATE_PURGE_JOB(
             audit_trail_type =>
                   DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
   audit_trail_purge_interval => 24,
       audit_trail_purge_name => 'daily_purge_job',
      use_last_arch_timestamp => TRUE);
END;
```

Die View UNIFIED_AUDIT_TRAIL

- Diese View ist ein Monster mit 94 Spalten
  - von denen sich allerdings 42 auf Extra-Optionen wie Database Vault, Label Security, FGA und Real Application Testing beziehen.
- Hier ein Beispiel für eine "kleine" Auswahl (neue Spalten in rot)

```
CREATE OR REPLACE VIEW trail AS
SELECT os_username,userhost,dbusername,
       REGEXP_SUBSTR(authentication_type, '[^=();]*',1,24) ip_addr,
       TO_CHAR(event_timestamp,'dd.mm.rr hh24:mi:ss') datum,
       action_name, return_code, sql_text, sessionid, scn,
       dblink_info, client_program_name,dbproxy_username,
       external_userid, client_identifier,
       object_schema,object_name,system_privilege_used,
       system_privilege, object_privileges, role,
       unified_audit_policies, audit_option
FROM unified_audit_trail
ORDER BY event_timestamp DESC;
```

Vordefinierte oder eigene Policies nutzen ?

- **Von den vordefinierten Policies sind nur 2 aktiviert**
  - **ORA_CIS_RECOMMENDATIONS (Empfehlungen des Center for Internet Security)**
  - **ORA_LOGON_FAILURES**
  - **ORA_RAS_POLICY_MGMT,ORA_RAS_SESSION_MGMT (betrifft das Feature *Real Application Security,* eine Erweiterung der Virtual Private Database, die nur in der EE erhältlich ist)**
  - **ORA_DATABASE_PARAMETER**
  - **ORA_ACCOUNT_MGMT**
  - **ORA_SECURECONFIG**

- **Oracle empfiehlt, möglichst wenige Policies gleichzeitig zu aktivieren, um den Overhead klein zu halten.**

Session Audit

- **Die folgende View liefert gescheiterte Login-Versuche:**

```
CREATE OR REPLACE VIEW failed_logins AS
SELECT os_username,userhost,dbusername,datum,
       client_program_name,return_code
FROM trail
WHERE unified_audit_policies = 'ORA_LOGON_FAILURES';
```

- **Alternativ kann man LOGIN und LOGOFF auditieren wie unter 11g**

```
NOAUDIT POLICY ora_logon_failures;
CREATE AUDIT POLICY logins_11g_pol ACTIONS LOGON, LOGOFF;
AUDIT POLICY logins_11g_pol;
```

- **Wichtig:**
  - **Wenn der Parameter audit_trail auf NONE steht, werden ungültige Login-Versuche bei älteren PSU-Patch-Leveln nicht auditiert.**

Session Audit (f)

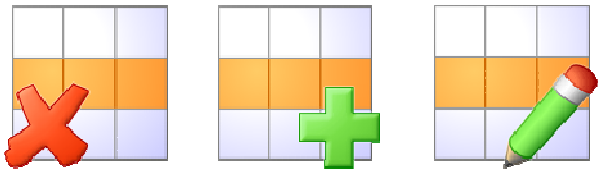
- Man kann natürlich (ggf. nach Absprache mit dem Betriebsrat) auch gezielt die Login-Versuche einzelner User auditieren, z.B.
  - `CREATE AUDIT POLICY log_scott_pol ACTIONS LOGON, LOGOFF;`
  - `AUDIT POLICY log_scott_pol BY scott WHENEVER SUCCESSFUL;`

- Wenn man eine Policy löschen will, muss man vorher beim NOAUDIT dieselben Klauseln verwenden
  - `NOAUDIT POLICY log_scott_pol;`
  - `DROP AUDIT POLICY log_scott_pol;`
    `ORA-46361: Audit-Policy kann nicht gelöscht werden, weil sie aktuell aktiviert ist.`
  - `NOAUDIT POLICY log_scott_pol BY scott WHENEVER SUCCESSFUL;`
  - `DROP AUDIT POLICY log_scott_pol;`
    `Audit POLICY gelöscht.`

DML-Aktionen auditieren

- **Diese Policy überwacht Änderungen an der Tabelle KUNDENDATEN**

```
CREATE AUDIT POLICY kunden_dml_pol
  ACTIONS INSERT ON kunden.kundendaten,
          UPDATE ON kunden.kundendaten,
          DELETE ON kunden.kundendaten;
AUDIT POLICY kunden_dml_pol;
```

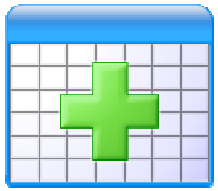


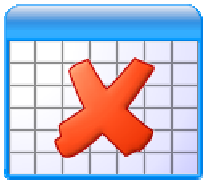
- **Änderung an Objekt-Audit-Optionen greifen sofort, nicht erst in der nächsten Session wie Privilege- und Statement-Audit-Optionen !!**
- **Auswertung mit**

```
CREATE OR REPLACE VIEW kundendaten_dml AS
SELECT os_username,userhost, dbusername,client_program_name,
       datum, action_name,return_code,sql_text
FROM trail
WHERE object_name = 'KUNDENDATEN'
ORDER BY datum DESC;
```

DDL-Aktionen auditieren

- Die Policy ORA_SECURECONFIG auditiert nur einen Teil der Aktionen, die unseren DBA interessieren.
- Es fehlen z.B.
  - Audit von Triggern, auch mit ANY-Rechten
  - Audit von Views, Sequenzen, Indizes, Types, Materialized Views und Materialized View Logs
  - Audit von TRUNCATE TABLE-Aktionen
  - Audit von DDL-Aktionen im eigenen Schema





- Er beschließt, die Policy ORA_SECURECONFIG zu deaktivieren und sich eigene Policies zu erstellen.
  - `NOAUDIT POLICY ora_secureconfig;`

Trigger →

DDL-Aktionen auditieren (f)

- **Diese Policy soll die wichtigsten DDL-Aktionen der beiden User kunden und kd_verw aufzeichnen.**
  - **Eigentlich sollten in der neuen Syntax keine Abkürzungen mehr gelten, aber PROCEDURE steht auch für FUNCTION und PACKAGE.**

```
CREATE AUDIT POLICY ddl_pol
  ACTIONS  CREATE TABLE, DROP TABLE,
           ALTER TABLE, TRUNCATE TABLE,
           CREATE PROCEDURE,DROP PROCEDURE,
           CREATE VIEW, DROP VIEW, ALTER VIEW,
           CREATE INDEX, DROP INDEX,
           CREATE TRIGGER,ALTER TRIGGER,DROP TRIGGER
WHEN 'SYS_CONTEXT(''USERENV'',''CURRENT_SCHEMA'')
      IN (''KUNDEN'', ''KD_VERW'')'
EVALUATE PER STATEMENT;
AUDIT POLICY ddl_pol;
```

DDL-Aktionen auditieren (f)

- Durch das Löschen der Bedingung kann man die Policy auf alle Datenbankuser erweitern.
  - `ALTER AUDIT POLICY ddl_pol CONDITION DROP;`
- Besser: Policy deaktivieren und neu erstellen

- Beispiel für die Auswertung:

```
CREATE OR REPLACE VIEW ddl_audit
AS
SELECT os_username, userhost, dbusername, object_schema
       client_program_name, ip_addr, datum, action_name,
       return_code, sessionid, scn, object_name,
       system_privilege_used, sql_text
FROM trail
ORDER BY datum DESC;
```

Audit administrativer Aktionen

- Auch bei den administrativen Aktionen vermisst der DBA einige Punkte in der Policy ORA_SECURECONFIG
  - Erstellen, Löschen und Verändern von Tablespaces
  - Erstellung von PFILES
  - Rechtevergabe unter den "Normal-Usern", wenn z.B. der User kd_verw dem User Kunden Objektrechte verleiht bzw. entzieht.
- Andere Audit-Optionen der Default-Policy wie z.B. `TRANSLATE ANY SQL`, `ALTER PLUGGABLE DATABASE`, `ADMINISTER KEY MANAGEMENT` sind für ihn nicht relevant.
- Nachdem er die ORA_SECURECONFIG-Policy deaktiviert hat, braucht noch eine Policy für Änderungen an der Datenbank, User- und Rechteverwaltung und andere nicht alltägliche Aktionen.

Audit administrativer Aktionen (f)

```
CREATE AUDIT POLICY admin_aktionen_pol
PRIVILEGES CREATE EXTERNAL JOB, CREATE JOB, DROP PUBLIC SYNONYM,
           CREATE PUBLIC SYNONYM, PURGE DBA_RECYCLEBIN
ACTIONS ALTER DATABASE, ALTER SYSTEM, CREATE PFILE,
        CREATE TABLESPACE, ALTER TABLESPACE, DROP TABLESPACE,
        CREATE USER, ALTER USER, DROP USER, GRANT, REVOKE,
        CREATE ROLE, ALTER ROLE, DROP ROLE, SET ROLE,
        CREATE PROFILE, ALTER PROFILE, DROP PROFILE,
        CREATE DATABASE LINK, ALTER DATABASE LINK,
        DROP DATABASE LINK, CREATE DIRECTORY, DROP DIRECTORY;
AUDIT POLICY admin_aktionen_pol;
```

- **Wichtig:**
  - **Alle Statement und Privilege-Audit-Optionen bleiben für die Dauer der Session aktiv.**
  - **Änderungen an den Audit-Optionen greifen erst mit Beginn der nächsten Session.**

Policy für das Überwachen einzelner User

- Die meisten Aktionen einzelner User sollten jetzt eigentlich durch die Policies ADMIN_AKTIONEN_POL und DDL_POL abgedeckt sein, aber der DBA startet trotzdem mal eine pauschale Überwachung für den nacht-aktiven Werkstudenten auf der Testinstanz.

```
CREATE AUDIT POLICY audit_user_pol ACTIONS ALL;
AUDIT POLICY audit_user_pol BY kd_verw;
```

- Problem:
  - Was jetzt alles auditiert wird, erfährt man nicht aus der Doku oder der View UNIFIED_AUDIT_TRAIL.
  - Ein Blick in AUDIT_ACTIONS oder AUDITABLE_SYSTEM_ACTIONS hilft hier weiter (und lässt nichts Gutes ahnen).
  - Ein einfaches `SELECT * FROM all_users` liefert je nach verwendetem Tool 10 (SQL*Plus) oder 35 Zeilen (SQL Developer) im Audit.

Policy für das Überwachen einzelner User(f)

- **Überraschungen beim Deaktivieren und Droppen der Policy**
  - **In SE2-Datenbanken mit älteren PSU-Patch-Leveln liess sich die Policy z.T. nur deaktivieren und droppen, wenn der User gedroppt wurde .**
  - **Deaktivieren funktioniert nur, wenn der User keine Session mehr offen hat.**

- **Wenn man alle Aktionen eines Users auditieren will, sollte man die Policy besser nach den Rechten des Users ausrichten.**
  - **So einfach geht's leider nicht**

```
CREATE AUDIT POLICY audit_user_pol PRIVILEGES ALL;
ERROR at line 1:
ORA-46355: missing or invalid privilege audit option.
```

Privilege-Audit einzelner User

- **Hier eine genial einfache Lösung von Marco Patzwahl (leicht gekürzt) :**

```
WITH s AS (SELECT 'SCOTT' AS uname FROM dual)
SELECT 'CREATE AUDIT POLICY '||LOWER(s.uname)||'_priv_capt ' FROM s
UNION -- System-Rechte
SELECT 'PRIVILEGES '||(LISTAGG(privilege,','||CHR(10))
               WITHIN GROUP (ORDER BY privilege))||';'
FROM s, dba_sys_privs WHERE grantee = s.uname
UNION ALL -- Objektrechte
SELECT 'ALTER AUDIT POLICY '||LOWER(s.uname)||'_priv_capt ADD ACTIONS '||
         privilege||
       CASE type WHEN 'DIRECTORY' THEN ' ON DIRECTORY ' ELSE ' ON ' END
       ||owner||'.'||table_name||';'
FROM s, dba_tab_privs WHERE grantee = s.uname
UNION ALL -- Rechte über Rollen (nur erste Hierachieform)
SELECT 'ALTER AUDIT POLICY '||LOWER(s.uname)||
         '_priv_capt ADD ROLES '||granted_role||';'
FROM s, dba_role_privs WHERE grantee = s.uname
UNION ALL
SELECT 'AUDIT POLICY '||LOWER(s.uname)||'_priv_capt BY '||s.uname||';' FROM s;
```

Fazit

- In vieler Hinsicht ist das neue Unified Auditing ein absoluter Fortschritt, aber es gibt einiges zu beachten:
  - Wenn die zu auditierende Instanz häufiger crasht oder der Server gerne mal neu gestartet wird, sollte man das Audit besser auf sofortiges Schreiben (immediate write) umstellen.
  - Die Syntax ist zwar flexibler und etwas übersichtlicher geworden, aber abseits der Standard-Beispiele findet man nicht sehr viel und ist beim Konfigurieren des Audits auf eigene, gründliche Tests angewiesen.
  - Unbedingt immer die neuesten PSU-Patches einspielen !
- Wer schon auf 12c umgestiegen ist, sollte gleich auf Unified Auditing setzen.
- Ansonsten gilt (meiner Meinung nach): Auf 12.2 warten, Tee trinken und schon mal anfangen, zu testen !