# Top 5 Features in Oracle® 12c

**Marco Patzwahl**
**MuniQSoft GmbH**
**Unterhaching**

**Schlüsselworte:**

Features, Spaltenlänge, IDENTITY, Unified Auditing, Datendateien, Pluggable Database, Container Datenbank

## Einleitung

Oracle 12c verfügt über mehr als 300 Neuerungen. Im folgenden Artikel wollen wir Ihnen fünf ausgewählte vorstellen.

### Platz 5: Neue Spaltenlänge für VARCHAR2 Spalten

Ab Version 12.1 kann für VARCHAR2 und Raw die Länge 32767 Bytes verwendet werden. Dazu muss die DB einmalig umgestellt werden.

### Umstellung einer DB auf neue Spaltenlänge

1. Schließen Sie die Datenbank.

```
shutdown immediate
```

2. DB starten im Upgrade Modus

```
startup upgrade
```

3. Einstellung von MAX_STRING_SIZE ändern

```
ALTER SYSTEM SET MAX_STRING_SIZE =extended;
```

4. Starten Sie das Skript

```
@?/rdbms/admin/utl32k.sql
```

5. DB neu starten

```
shutdown immediate
startup
```

### Neue Länge für VARCHAR2(32767)

Nun können Sie die neue Spaltenlänge nutzen:
Beispiel:

```
CREATE TABLE t (
id              NUMBER,
text            VARCHAR2(32767));
INSERT INTO t (text) VALUES(lpad('#',32000,'X'));
SELECT length(text) FROM t;
```

=> 32000

### Nebenwirkungen der neuen Länge

Ein Index auf einer 32K Spalte:

```
CREATE INDEX i_text_ix ON t(text);
```

Endet mit einem
ORA-01450: Maximale Schlüssellänge (6398) überschritten

**Platz 4: IDENTITY Spalte**
Sie können auch einen automatischen Sequenz-Generator zum Füllen von PK/UK Spalten verwenden

```
CREATE TABLE t (id NUMBER GENERATED AS IDENTITY, text VARCHAR2(10));INSERT
INTO t (text) VALUES ('X');

CREATE TABLE t (id NUMBER GENERATED BY DEFAULT AS IDENTITY (START WITH 100
INCREMENT BY 10));
```

**Identity Optionen**
Im Prinzip können die gleichen Optionen wie bei einer Sequenz verwendet werden:

```
...( START WITH ( <int>|LIMIT VALUE )|INCREMENT BY <int>| ( MAXVALUE integer
| NOMAXVALUE ) | ( MINVALUE integer | NOMINVALUE ) | ( CYCLE | NOCYCLE ) |
( CACHE integer | NOCACHE ) | ( ORDER | NOORDER )
```

Zusätzlich gibt es noch:
ALWAYS: Spalte darf nur durch Identity gefüllt werden. Manuell bei Insert oder Update ist nicht erlaubt
BY DEFAULT [ON NULL]: Spalte wird per Default durch Identity gefüllt, manuelles Füllen ist jedoch möglich. ON NULL: Wenn ein Insert für die Spalte Null ergibt, wird durch Identity gefüllt.

**Tabellenänderungen: IDENTITY Spalte**
Wie funktioniert es ?
Oracle legt eine Sequenz im Schema der Tabelle an (z.B. mit Namen ISEQ$$_12345 und als Default mit Cache=20)
Welche Tabellen verwenden Identity Spalten :

```
SELECT owner,table_name,has_identity
FROM all_tables
WHERE owner='MARCO';

SELECT owner,table_name, column_name,identity_column
FROM all_tab_columns
WHERE owner='MARCO'
```

**Identity Wartung:**
Bestehende Identity löschen (Spalte bleibt erhalten):

```
ALTER TABLE <schema>.<TABLE> MODIFY (<ident_col>  DROP IDENTITY);
```

Beispiel:

```
ALTER TABLE t MODIFY (id DROP IDENTITY);
```

Identity nachträglich (incl. Spalte) hinzufügen:

```
ALTER TABLE T ADD (ID_NEU NUMBER GENERATED AS IDENTITY (START WITH 100));
```

Hinweis: Eine Identity nachträglich zu einer existierenden Spalte hinzuzufügen ist nicht vorgesehen!

**Platz 3: Unified Auditing**
Das Feature UNIFIED AUDITING fasst sämtliche Audit-Informationen an einer einzigen Stelle zusammen.
Dadurch ergeben sich einige Vorteile
Keine Parametereinstellung mehr notwendig
Bessere Übersicht und einfachere Sichtung der Audit-Informationen
Bessere Performance, was die Überwachung betrifft
Ermittlung, ob Option bereits eingeschalten (TRUE oder FALSE):

```
SELECT * FROM v$option WHERE parameter =  'Unified Auditing';
```

**Unified Auditing (f)**
Bei FALSE muss die Nutzung explizit noch aktiviert werden
Dazu Instanz und Listener herunterfahren
Unter Unix Wechsel in $ORACLE_HOME/rdbms/lib, dann
`make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME`
Unter Windows in %ORACLE_HOME%\bin folgende Datei umbenennen
`orauniaud12.dll.dbl in orauniaud12.dll`
Listener und Instanz wieder starten

**Unified Auditing Speichermethoden**
Der Initialisierungsparameter UNIFIED_AUDIT_SGA_ QUEUE_SIZE regelt die Speichermenge, die der Hauptspeicher für die Audit Einträge zur Verfügung stellt
Speichergröße 1-30MB
Benutzer die Unified Auditing administrieren wollen, benötigen das Recht AUDIT SYSTEM oder die Rolle AUDIT ADMIN
Die Auditdaten werden zuerst gecached und periodisch in Tabellen gespeichert
Bei einem Shutdown abort könnten jedoch Einträge verloren gehen. Deswegen unterstützt Oracle zwei Modi:

- Immediate (sofort schreiben)
- Queued Write (verzögert schreiben)

```
BEGIN /* Immediate Write */
DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY(
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
DBMS_AUDIT_MGMT.AUDIT_TRAIL_WRITE_MODE,
DBMS_AUDIT_MGMT.AUDIT_TRAIL_IMMEDIATE_WRITE);
END;

BEGIN /* Queued Write */
DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY(
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
DBMS_AUDIT_MGMT.AUDIT_TRAIL_WRITE_MODE,
DBMS_AUDIT_MGMT.AUDIT_TRAIL_QUEUED_WRITE);
END;
```

Aktuell im Speicher befindliche Audits sofort auf Disk schreiben:
`EXEC DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL;`

**Unified Auditing Syntax:**
Syntax:

```
CREATE AUDIT POLICY policy
[ privilege_audit_clause ]
[ standard_or_component_clause ]
[ role_audit_clause ]
[ WHEN 'audit_condition'
EVALUATE PER
{ STATEMENT | SESSION | INSTANCE } ] [ CONTAINER = { ALL | CURRENT } ] ;
```

EVALUATE PER …
STATEMENT: Für jedes ausgeführte Statement wird ein Auditeintrag erstellt
SESSION: Innerhalb wird nur einmal für ein ausgeführtes Statement ein Auditeintrag erzeugt
INSTANCE: Solange die Instanz läuft, wird Vorgehen nur einmal aufgezeichnet
CONTAINER =
    ALL (Alle Container einer Pluggable Database)
    CURRENT (nur der aktuelle Container)

**Beispiele zur WHEN Klausel**
SELECT ANY TABLE und CREATE VIEW für zwei Benutzer überwachen:
```
CREATE AUDIT POLICY osusers_tab_view_pol
PRIVILEGES SELECT ANY TABLE, CREATE VIEW WHEN q'!SYS_CONTEXT ('USERENV',
'OS_USER') IN ('MARCO', 'HANS')!'
EVALUATE PER SESSION;
```
Alle Anmeldungen durch SQL*Plus überwachen:
```
CREATE AUDIT POLICY sqlplus_logon_pol
ACTIONS LOGON WHEN q'!INSTR(UPPER(SYS_CONTEXT('USERENV',
'CLIENT_PROGRAM_NAME')), 'SQLPLUS') > 0!'
EVALUATE PER SESSION;
```

**Unified Auditing Beispiele**
Beispiele:
```
CREATE AUDIT POLICY dml_poli ACTIONS
DELETE on scott.emp,
INSERT on scott.emp,
UPDATE on scott.emp,
ALL    on scott.dept;
AUDIT POLICY dml_poli BY scott;
```

**Unified Auditing Beispiele für Privilegien**
```
CREATE AUDIT POLICY table_poli PRIVILEGES
CREATE ANY TABLE,
DROP ANY TABLE;
AUDIT POLICY table_poli BY scott;
```

**Interessante Privilegien zum Audit**
```
CREATE USER                          CREATE LIBRARY
ALTER USER                           CREATE PUBLIC SYNONYM
CREATE DATABASE LINK                 DROP TABLESPACE
ALTER DATABASE LINK                  DROP ANY TABLE
ALTER SESSION                        DROP USER
ALTER SYSTEM                         EXEMPT ACCESS POLICY
AUDIT ANY                            EXEMPT DDL REDACTION POLICY
AUDIT SYSTEM                         EXEMPT DML REDACTION POLICY
BECOME USER                          EXEMPT IDENTITY POLICY
CREATE ANY DIRECTORY                 EXEMPT REDACTION POLICY
CREATE ANY JOB                       EXPORT FULL DATABASE
CREATE EXTERNAL JOB                  GRANT ANY OBJECT PRIVILEGE
CREATE JOB                           GRANT ANY PRIVILEGE
```

**Unified Auditing Beispiele für Packages**
Beispiele:
Überwachen, ob SYS Rechte an UTL_FILE/UTL_TCP/UTL_SMTP vergibt:

```
CREATE AUDIT POLICY dbms_utl_grants ACTIONS
GRANT ON UTL_FILE,
GRANT ON UTL_TCP,
GRANT ON UTL_SMTP;
AUDIT POLICY dbms_utl_grants BY SYS;
```

**Lohnenswerte Packages für die Überwachung:**

| | |
|---|---|
| UTL_FILE | DBMS_SYS_SQL |
| UTL_SMTP | DBMS_FGA |
| APEX_MAIL | DBMS_RLS |
| UTL_HTTP | DBMS_CRYPTO |
| UTL_TCP | DBMS_JOB |
| UTL_INADDR | DBMS_SCHEDULER |
| DBMS_SQL | |

**Audits abändern**
Eine bestehende AUDIT Policy kann erweitert oder reduziert werden mittels:

```
ALTER AUDIT POLICY <policy_name>
[ADD [privilege_audit_clause] [action_audit_clause] [role_audit_clause]]
[DROP [privilege_audit_clause] [action_audit_clause] [role_audit_clause]]
[CONDITION {DROP | audit_condition
EVALUATE PER {STATEMENT|SESSION|INSTANCE}}]
```

**Beispiele zu Audits abändern**
Zur Policy tab_audpol wird die Überwachung des Rechners "Bluemchen" hinzugefügt, wenn er einen Insert auf die dept Tabelle durchführt:

```
ALTER AUDIT POLICY tab_audpol ADD ACTIONS INSERT ON SCOTT.DEPT CONDITION
q'!SYS_CONTEXT('HOST', 'MY_HOST') = 'BLUEMCHEN'!' EVALUATE PER SESSION;
```

**Audit auswerten**

```
SELECT TO_CHAR(EVENT_TIMESTAMP,'DD.MM.YY Hh24:MI:SS') AS
EVENT_TIMESTAMP,OS_USERNAME,DBUSERNAME,substr(client_program_name,1,30)
client_prg, action_name, object_name, sql_text, system_privilege_used
FROM UNIFIED_AUDIT_TRAIL
order by 1 desc;
```

**Platz 2: Datendateien verschieben**

```
ALTER DATABASE MOVE DATAFILE ( 'filename' | 'ASM_filename' | file_number )
[ TO ( 'filename' | 'ASM_filename' ) ]
[ REUSE ] /* vorhandene Datei überschreiben */
[ KEEP ]; /* alte Version behalten */
```

Beispiel:

```
ALTER DATABASE MOVE DATAFILE 4 TO '/u01/app/oracle/oradata/user02.dbf'
KEEP;
```

**Platz 1: Pluggable Database**

**Generelles zu Pluggable Database**
Pluggable Database ist eine Containerdatenbank (CDB) für 0, 1 oder mehrere Pluggable Databases (PDB)
Jede CDB besteht aus

- Genau einem ROOT (CDB$ROOT)
  Hier stehen die Metadaten und die Standard Benutzer (die in jedem Container bekannt sind.)
- Genau einer Seed PDB (PDB$SEED)
  Dies ist ein Template zur Erstellung von neuen PDBs. Objekte in PDB$SEED dürfen NICHT verändert werden
- Null-n PDBs:
  Benutzerdefinierter und erstellter Container für Daten oder Code (evtl. eigens angelegt für verschiedene Applikationen)

Mehrere PDBs können in einer einzelnen CDB liegen. Sie können PDBs aus einer CDB entfernen und in eine andere verschieben. Sie können PDBs klonen und PDB ist kompatibel zu allen Oracle Optionen und RAC

**PDB Verwaltungsviews**
Neue Data Dictionary Views (z.B. CDB_PDBs)
Viele Views haben neue Spalte: Con_ID (Container ID)
Alte Data Dictionary View (DBA_, All_, and User_) wurden ergänzt um solche mit dem CDB_ Prefix.
Die CDB_Views besitzen die neue Spalte Con_ID.
Test, ob die angebundene DB eine CDB ist:

```
SELECT count(*) FROM v$pdbs;
```

0 => Nein keine PDB, sondern normale DB
1-n => CDB mit n PDB Datenbanken oder …

```
SELECT cdb FROM V$database;
```

=> YES

**V$PDB / CDB_PDBS**
Welche Container gibt es ?

```
SELECT con_id,name,open_mode,restricted, open_time
FROM v$pdbs;
SELECT pdb_id,con_id,pdb_name,dbid,status
FROM CDB_PDBS;
```

**V$SESSION**
Wer ist in welchem Container angemeldet ?

```
SELECT con_id,sid,serial#,username,program
FROM v$session ORDER BY 1 DESC;
```

**V$DATAFILE**
Zu welchem Tablespace gehören Dateien im Container 2?

```
SELECT tablespace_name,file_name
FROM CDB_DATA_FILES WHERE con_id=2;
```

**Aktuelle PDB anzeigen/wechseln**
Container in der Session wechseln:

```
ALTER SESSION SET CONTAINER=pdb1;
```

CDB$ROOT = Root Container

PDB$SEED = Seed Container
In welchem Container befindet sich die Session?

```
SELECT Sys_Context('Userenv', 'Con_Name') "current container"
FROM dual;
```

Alle Container in SQL*Plus anzeigen:

```
show pdbs
```

**Wechsel des Containers:**
Dadurch werden kein Login Trigger gezündet.
Workaround: AFTER DDL Trigger mit ALTER SESSION SET CONTAINER schreiben
Package Statuswerte werden nicht über die Containergrenze hinweg zur Verfügung gestellt
Transaktionen können nicht über mehrere Container gehandlet werden. Wenn eine Transaktion beginnt und der Container gewechselt wird, kann dort keine DML, Commit oder Rollback Anweisung durchgeführt werden, solange nicht in den Ursprungscontainer gewechselt wurde.

**Benutzer anlegen in CDB**
Benutzer , die in einer CDB angelegt werden, müssen mit dem Prefix c## oder C## beginnen
Dadurch erhofft sich Oracle weniger Kollisionen mit bestehenden Benutzern in einer PDB
Oracle selbst verwendet einen undokumentierten Parameter, um Benutzer (z.B. SYSTEM) für alle PDBs verfügbar zu machen:

```
ALTER SESSION SET "_oracle_script"=true;
ALTER SESSION SET "_common_user_prefix"='';
CREATE USER marco IDENTIFIED BY marco;
Benutzer wurde erstellt.
```

Beachten Sie bitte, dass das kein von Oracle unterstütztes Verfahren ist!

**Globalen Benutzer anlegen in CDB:**
Beispiel:

```
CREATE USER c##comm_user IDENTIFIED BY comm_pwd
DEFAULT TABLESPACE users QUOTA 32M ON users
TEMPORARY TABLESPACE temp
PROFILE comm_prof;
```

Hinweise:
Der Tablespace Users und Temp und das Profile comm_prof muss in allen Containern, die zur DB gehören, existieren
In einer Nicht-Pluggable DB ist der Prefix C## für Benutzer nicht erlaubt!

**Anmelden am Container**
Bei zwei Containern pdb1 und pdb2:

```
SQL> connect scott/TIGER@172.30.30.1/pdb1
SQL> connect scott/TIGER@172.30.30.1/pdb2
```

Root DB (o12c):

```
SQL> connect scott/TIGER@172.30.30.1/o12c
```

**Automatische Anmeldung an Container**
Dies funktioniert nur, wenn der Benutzer in allen Containern verfügbar ist (C##<USER>)

```
CREATE OR REPLACE TRIGGER system.switch_to_pdb
AFTER LOGON ON DATABASE BEGIN
IF Sys_Context('USERENV','CURRENT_USER')='C##SH'
THEN
execute immediate 'alter session set container=pd1';
END IF;
```

```
END;
/
```

**Database Links**

Sie können mittels Database Links auf die verschiedenen Container zugreifen:

```
CREATE DATABASE LINK pdb1 CONNECT TO scott IDENTIFIED BY TIGER USING
'172.30.30.30/pdb1';
CREATE DATABASE LINK pdb2 CONNECT TO scott IDENTIFIED BY TIGER USING
'172.30.30.30/pdb2';
SELECT * from emp@pdb1;
SELECT * from emp@pdb2;
```

Kontaktadresse:

Marco Patzwahl
MuniQSoft GmbH
Grünwalder Weg 13a
D-82008 Unterhaching

Telefon: +49 (0) 89-67 90 90 40
Fax: +49 (0) 89-67 90 90 50
E-Mail info@muniqsoft.de
Internet: www.muniqsoft.de