ORACLE 12C NEW FEATURE

A Resource Guide

NOV 1, 2016

TECHGOEASY.COM

MULTITENANT ARCHITECTURE AND PLUGGABLE DATABASE

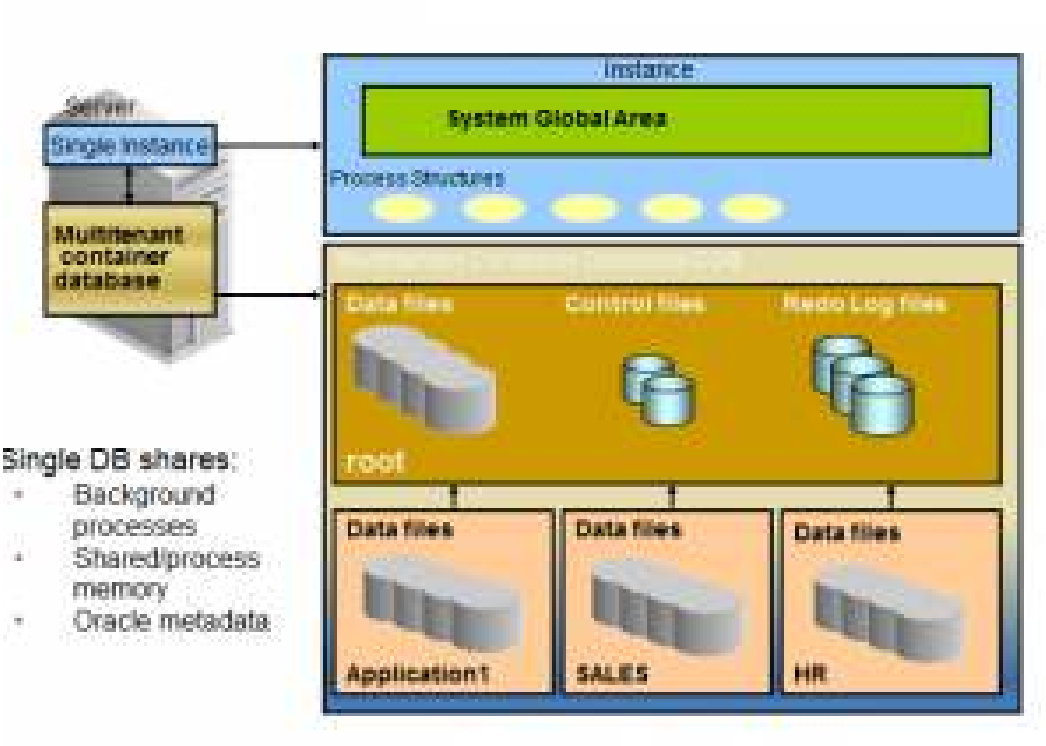
Why Multitenant Architecture introduced with 12c?

Many Oracle customers have large numbers of applications built on Oracle RDBMS. They do not use a significant percentage of the hardware on which they are deployed. Customers have instance and storage overhead preventing large numbers of databases from being placed on the same physical and storage server. They are not generally complex to require 100% of the attention of a full time administrator and they do require significant time to patch or upgrade all applications.

Multitenant Architecture is the answer to all these problems. It basically provides a cloud of Oracle database.

What is Multitenant Architecture?

Multiple tenants share same resources on a mutual benefit for different purposes at a very broad level. The same applies to Oracle Database where Multiple Databases share a single instance of resources aiming for different purposes on the same Server. This Oracle Database which is built on Multitenant foundation is called Container Database(CDB), and each container(tenant) residing inside is called Pluggable Database (PDB, Container).



Features

1) All the Pluggable database shares the same background process, Shared /process memory, Oracle metadata

2) All the Pluggable database shares the redo log file, control files and undo tablespace

3) There are two types of containers in **Multitenant Architecture**–The root container:

- The first container created at CDB creation
- Mandatory
- Oracle system-supplied common objects and metadata
- Oracle system-supplied common users and roles

–Pluggable database containers (PDBs):

- A container for an application:

–Tablespaces (permanent and temporary)

–Schemas / objects / privileges

–Created / cloned / unplugged / plugged

- Particular seed PDB:

–PDB$SEED provides fast provisioning of a new PDB

4) There is a Limit of 253 PDBs in a CDB including the seed

5) There is a Limit of 1024 services in a CDB

6) Every PDB has its own set of SYSTEM/SYSAUX/TEMP tablespaces, and also includes the Sample User data such as SCOTT etc.

7) ENABLE_PLUGGABE_DATABASE initialization parameter specifies If a particular Database is CDB or Non-CDB

Benefits

1)It operates multiple database in a centrally managed platform to lower costs. It basically consolidates all the databases. It means less instance overhead and less storage utilization. Till 11g, we have databases deployed across multiple Small Physical machines on various platforms. It could be waste of resources having a database on each machine dedicatedly rather keeping them all on a single powerful machine. 12c Multitenant architecture consolidates all Databases onto a Single powerful chip and a Single Oracle Instance

2) It allows central management and database administration of multiple database. Backup and recovery, patching, upgrade becomes simpler

3) It is supported with Data guard and RAC, So Disaster recovery and High Availability can be achieved for multiple database at the same time

4) It reduces the DBA resource costs

SELECT IMPROVEMENTS

1) TOP-N FEATURE

A Top-N query allows us to retrieve the top or bottom N rows from an ordered set. Combining two Top-N queries gives you the ability to page through an ordered set

Example:

```
SELECT value FROM mytable ORDER BY value DESC

FETCH FIRST 10 ROWS ONLY;

select * from my_test order by name fetch first 3 rows only;
```

Pre 12c we need to write queries like this to achieve it

```
select … from (select … from … order by …) where rownum <= 20
```

The optimizer uses analytic under the cover to make it work

Pagination can also happen with this feature with the use offset syntax

```
– offset 10 rows fetch first 10 rows only

select * from my_test order by id
offset 10 rows fetch next 10 rows only;
– offset 10 rows fetch first 0.1 percent rows only

select * from my_test order by id

offset 10 rows first 0.1 percent rows only

– offset 10 rows fetch first 0.1 percent rows with ties
```

Restriction
1) If you have a SELECT statement with FOR UPDATE, you can't use it.
2) The SELECT statement can't CURRVAL or NEXTVAL of sequences
3) If the query of the Materialized Views has this clause, then you can't do an incremental refresh of that MV

2) EXTENDED OUTER JOIN

Oracle 12c extends the power of outer join to make it more useful
Now you can use outer join for multiple table and column

```
select *

from x, y, z

where x.uid = y.uid

and x.uid = z.uid(+)

and y.uid = z.uid(+);
```

The above query will give error pre 12c but execute successfully in 12c

3) LATERAL INLINE VIEWS, CROSS APPLY AND OUTER APPLY EXTENSION

Oracle 12c introduced the LATERAL inline view syntax, as well as CROSS APPLY and OUTER APPLY joins into the SELECT syntax.

LATERAL INLINE VIEW

Normally, it is not possible to reference tables outside of an inline view definition.

```
SELECT dept_no, emp_no

FROM depts d,

(SELECT emp_no

FROM emp e

WHERE e.dept_no = d.dept_no);WHERE e.dept_no = d.dept_no)

*

ERROR at line 5:

ORA-00904: "D"."DEPT_NO": invalid identifier
```

A LATERAL inline view allows you to reference the table on the left of the inline view definition in the FROM clause, allowing the inline view to be correlated.

```
SELECT dept_no, emp_no

FROM depts d,

Lateral (SELECT emp_no

FROM emp e

WHERE e.dept_no = d.dept_no);
```

CROSS APPLY

It works like normal join between two tables. The CROSS APPLY join is a variant of the ANSI CROSS JOIN. It returns all rows from the left hand table, where at least one row is returned by the table reference or collection expression on the right. The right side of the APPLY can reference columns in the FROM clause to the left

```
select * from test1 a cross apply

(select *

from test2 b

where b.uid=a.uid);
```

OUTER APPLY

It works like left outer join between tables. The usage is similar to the CROSS APPLY join, but it returns all rows from the table on the left side of the join. If the right side of the join returns no rows, the corresponding columns in the output contain Nulls.

```
select * from test1 a outer apply

(select *

from test2 b

where b.uid=a.uid);
```

If you look at the explain plan for each of these, then these are normal oracle joins only

4) WITH CLAUSE IMPROVEMENT

In Oracle 12c, we can declare PL/SQL functions in the WITH Clause of a select statement and use it as an ordinary function. Using this construct results in better performance as compared with schema-level functions

```
Example:

WITH

FUNCTION add_func_test(n IN NUMBER) RETURN NUMBER IS

BEGIN

RETURN n+1;

END;

SELECT add_func_test(5)

FROM dual;
```

5) CASCADE FOR TRUNCATE AND EXCHANGE PARTITION.

With Oracle Database 12c, The TRUNCATE can be executed with CASCADE option which will also delete the child records.

When you truncate a parent table with child tables,
you get:
ORA-02266: unique/primary keys in table referenced by enabled foreign keys
In Oracle 12c, you can use:
truncate table <Parent> cascade;
Must have defined the FK as ON DELETE CASCADE.
Otherwise ORA-14705: unique or primary keys referenced by enabled foreign keys in table will result

6)EXPAND_SQL_TEXT PROCEDURE

New procedure EXPAND_SQL_TEXT of package DBMS_UTILITY helps to recursively replace any view references in the input SQL query with the corresponding view subquery.

What if you need to analyze following query
select * from test3

With new procedure it's piece of cake

```
set serveroutput on

declare

v_in_view clob := 'select * from TEST3';

v_out_view clob;

begin

dbms_utility.expand_sql_text(v_in_view, v_out_view);

dbms_output.put_line(v_out_view);

end;

/anonymous block completed

<view query will come here>
```

7) SQL SUPPORT IN RMAN

In Oracle 12C release 1 most of SQL and PL/SQL commands are supported in rman. We don't require any SQL prefix or quotes. DESCRIBE is also supported

```
Pre 12c

RMAN> sql 'SELECT username,machine FROM v$session';

With 12c

RMAN> SELECT username,machine FROM v$session;
```

SCHEMA LEVEL NEW FEATURES IN 12C

1. SEQUENCE AS DEFAULT VALUE

With Oracle Database 12c, we can directly assign sequence nextval as a default value for a column, so you no longer need to create a trigger to populate the column with the next value of sequence, you just need to declare it with table definition. It is a sort of auto increment feature for a column in oracle just like MySQL

Example:

```
create sequence tech_test_seq start with 1 increment by 1 nocycle;

create table test

(

id number default tech_test_seq. nextval primary key

name varchar (30)

);
```

2). INVISIBLE COLUMN:

With 12c, you can make Invisible column. A Column defined as invisible, will not appear in generic queries (select * from). An Invisible Column need to be explicitly referred to in the SQL statement or condition. Also invisible column must be explicitly referred in INSERT statement to insert the database into invisible columns.

Example:

```
create table my_test

(

id number,

name varchar2(100),

dept varchar2(100),

password varchar2(100) INVISIBLE

age varchar2(100)

Firstname varchar2(100)

lastname varchar2(100)

);
```

Important points

The table will be created with password column invisible

Select * from my_test ;

It will show all the column except password

Describe will not show invisible column

To make column visible in sql, we can set set colinvisible on

To make column visible from invisible

ALTER TABLE my_test MODIFY (password visible);

we can create index on invisble column

3) MULTIPLE INDEXES ON THE SAME COLUMN

Before Oracle Database 12c, we could not have multiple indexes on a single column. In Oracle Database 12c a column may have multiple indexes but all should be of different types i.e. if B-tree is already defined then you can create bitmap index on the same column. Like a column may have B-Tree and Bitmap Index both. But, only one index will be used at a given time.

```
SQL> create table test (id number, name varchar2(100));

Table created.

SQL> create index test_idx1 on test(id);

Index created.

SQL> create index test_idx2 on test(id);

create index test_idx2_02 on test(id)

*

ERROR at line 1:

ORA-01408: such column list already indexed

SQL> create bitmap index test_idx2_02 on test(id) invisible;
```

```
Index created.
```

Here we could create the index as it has different type and it is also invisible

4) VARCHAR2 LENGTH UP TO 32767

Until Oracle 11g SQL, the maximum precision allowed for a string type column was 4000. In Oracle 12c, the precision has been increased up to 32767 bytes or 32K. The new string data types will be known as Extended String Types in Oracle 12c. The feature is controlled by an initialization parameter MAX_STRING_SIZE. Increasing the allotted size for these data types allows users to store more information in character data types before switching to large objects (LOBs).

Procedure to enable the feature

```
shutdown immediate

startup upgrade

alter system set max_string_size=EXTENDED scope=both;

@<ORACLE_HOME>/rdbms/admin/utl32k.sql

Shutdown immediate

Startup
```

5) IDENTITY COLUMNS

In Oracle Database 12c, we can define Table columns with SQL keyword IDENTITY

which is an American National Standards Institute (ANSI) SQL keyword. Which are auto-incremented at the time of insertion (like in MySQL).

```
Example:

create table test

(

id number generated as identity,

name varchar2(100),

email varchar2(100),

password varchar2(100)

firstname varchar2(100)

lastname varchar2(100));
```

6) SESSION SEQUENCES

With Oracle Database 12C new keywords SESSION, GLOBAL are available that can be specified during a sequence creation
CREATE SEQUENCE test_session_seq START WITH 1 INCREMENT BY 1 SESSION;
CREATE SEQUENCE test_global_seq START WITH 1 INCREMENT BY 1 GLOBAL;

Global	Session

creates standard sequence well known in previous release. This is the default.	creates new type session sequence, which is a special type of sequence that is specifically designed to be used with global temporary tables that have session visibility. Session sequence returns a unique range of sequence numbers only within a session, but not across sessions. Another difference is that session sequences are not persistent. If a session goes away, so does the state of the session sequences that were accessed during the session.

7) TEMPORARY UNDO

Before Oracle Database 12c, undo records of temporary tables used to be stored in undo tablespace. With the temporary undo feature in Oracle Database 12c, the undo records of temporary tables can now be stored in a temporary table instead of stored in undo tablespace. The main benefits of temporary undo are 1) Low undo tablespace usages 2) less redo data generation. For using this feature Compatibility parameter must be set to 12.0.0 or higher and TEMP_UNDO_ENABLED initialization parameter must be Enabled.

ADMINISTRATION NEW FEATURES

1) ONLINE RENAME/MOVE OF DATAFILES

A data file migration or renaming in Oracle database 12c R1 no longer requires a number of steps i.e. putting the tablespace in READ ONLY mode, followed by data file offline action. In 12c R1, a data file can be renamed or moved online simply using the ALTER DATABASE MOVE DATAFILE SQL statement. While the data file is being transferred, the end user can perform queries, DML and DDL tasks. Additionally, data files can be migrated between storages e.g. from non-ASM to ASM and vice versa.

Oracle Database 12c has provided a simple way to online renamed or moved data files by simply "ALTER DATABASE MOVE DATAFILE" command. Data files can also be migrated online from ASM to NON-ASM and NON-ASM to ASM easily now.

Examples:

Rename datafile:

SQL> ALTER DATABASE MOVE DATAFILE '/u01/app/oracle/TEST/oradata/applsysd.dbf' TO

'/u02/app/oracle/TEST/oradata/indx_01.dbf';

Move Datafile:

SQL> ALTER DATABASE MOVE DATAFILE '/u01/app/oracle/TEST/oradata/indx.dbf' TO

'/u02/app/oracle/TEST/oradata/indx.dbf';

NON-ASM to ASM:

SQL> ALTER DATABASE MOVE DATAFILE '/u01/app/oracle/TEST/oradata/indx.dbf' TO '+DATA01';

2) MOVE TABLE PARTITION TO DIFFERENT TABLESPACE ONLINE

Migration of a table partition or sub-partition to a different tablespace no longer requires a complex procedure in Oracle 12c R1. In a similar way to how a heap (non-partition) table online migration was achieved in the previous releases, a table partition or sub-partition can be moved to a different tablespace online or offline. When an ONLINE clause is specified, all DML operations can be performed without any interruption on the partition|sub-partition which is involved in the procedure. In contrast, no DML operations are allowed if the partition|sub-partition is moved offline. From Oracle 12c, it become very easy to move Table Partition to different tablespace and does not require complex steps

Example:

SQL> ALTER TABLE TEST_TABLE MOVE PARTITION TEST_PART1 TO

TABLESPACE <NEW tablespace>;

3) DDL LOGGING

By using the ENABLE_DDL_LOGGING initiation parameter in Oracle Database 12c, we can now log the DDL action into xml and log files to capture when the drop or create command was executed and by whom under the $ORACLE_BASE/diag/rdbms/DBNAME/log|ddl location. The parameter can be set at the database or session levels.

Example:

```
SQL> ALTER SYSTEM SET ENABLE_DDL_LOGGING=TRUE;
```

4) PGA_AGGREGATE_LIMIT PARAMETER

Oracle Database 12c has provided us a way to limit PGA by PGA_AGGREGATE_LIMIT parameter. Before Oracle Database 12c there was no option to limit and control the PGA size. Oracle will automatically abort the session that holds the most untenable PGA memory when PGA limits exceeds the defined value.

5) TURNING OFF REDO FOR DATA PUMP THE IMPORT

The new TRANSFORM option, DISABLE_ARCHIVE_LOGGING, to the impdp command line causes Oracle Data Pump to disable redo logging when loading data into tables and when creating indexes. This feature provides a great relief when importing large tables, and reduces the excessive redo generation, which results in quicker imports. This attribute applies to tables and indexes.

Example:

```
impdp directory=mydir dumpfile=mydmp.dmp logfile=mydmp.log tables=test

TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y table_exists_action=append

. . imported "TECH"."TEST"
```

6) MORE ONLINE OPERATIONS

```
DROP INDEX

DROP CONSTRAINT

SET UNUSED COLUMN

ALTER INDEX UNUSABLE

ALTER INDEX VISIBLE|INVISIBLE

ALTER TABLE MOVE (SUB)PARTITION)
```

SOME MORE INTERESTING FEATURES

1. **Temporal Validity**
   Temporal Validity is very interesting feature in Oracle 12C that provides ability to scan

effectively Gantt data
1) adds (one or more) “time dimension” to a table by using current columns or using columns automatically created by database
2) enable using simple SQL syntax to filter the columns to access only active data using Oracle flashback technology

2. **EBR Improvements**
   EBR was introduced in 11gR2. 12c gave more power to EBR. It removes some of the limitation. Now materialized view and types can be editioned
3. **Data redaction**

This is one of the top features in Oracle 12c. Data Redaction in simple terms means, masking of data. You can setup a Data Redaction policy, for example salary field in an Employee table can be masked. This is called redaction.

When you do a select * from employee, it will show that the Salary is masked.
The new data masking will use a package called DBMS_REDACT. It is the extension to the FGAC and VPD present in earlier versions.