EXAM OBJECTIVE : CREATE AND MAINTAIN BIGFILE TABLESPACES

In Oracle 10g, a new feature called a Bigfile tablespace enables a database to contain up to 8 **exabytes** of data. That's 8 million terabytes or three orders of magnitude over the current limit of 8 petabytes.

Note: 1 PB (petabyte) = 1024 TB (terabyte), 1 EB (exabytes) = 1024 PB = 1,048,576 TB = 2**60 bytes.

Bigfile tablespaces simplify database management by allowing you to create single-file tablespaces and letting you perform operations at the tablespace level rather than on the underlying datafile.

A bigfile tablespace (BFT) contains a single file that can have a very large size. The purpose of the bigfile tablespace feature is to increase storage and at the same time simplifying the management of datafiles and large databases.

GUIDELINES

- In Oracle 10g you can accommodate both smallfiles(datafiles created prior to Oracle 10g) and bigfiles.
- A bigfile when created should be locally managed and have its segment space management defined as automatic.
- Using a bigfile, you eliminate the need for multiple datafiles for a tablespace. There is only one datafile associated with a bigfile tablespace and any operations that could earlier be done on a datafile can now be performed on the bigfile tablespace
- When using bigfiles, think about the extent size before creating it. If the tablespace is really large (in terabytes), set UNIFORM SIZE otherwise leave it as the default AUTOALLOCATE.

PERSISTENT DATABASE PARAMETER – DEFAULT TABLESPACE TYPE.

A persistent database parameter can be set within the database that indicates the default tablespace type. The value will be used whenever a user does not explicitly specify the tablespace type when creating a tablespace. The following example shows how to set the default tablespace type for the database:

```
SQL> CREATE DATABASE
SET DEFAULT BIGFILE TABLESPACE
DATAFILE '        ' SIZE 200M
SYSAUX     DATAFILE '                ' SIZE 300M
```

....

SQL> ALTER DATABASE SET DEFAULT
BIGFILE TABLESPACE;

- The tablespace type for SYSTEM and SYSAUX tablespaces is always the same as the default tablespace types at the database creation type and cannot be overridden.
  CREATE DATABASE
  SET DEFAULT BIGFILE TABLESPACE
  DATAFILE '/disk1/system01.dbf' SIZE 200M
  SYSAUX DATAFILE '/disk1/sysaux1.dbf' SIZE 300M
  SMALLFILE DEFAULT TEMPORARY TABLESPACE stemp_tbs
  TEMPFILE '/disk2/stemp_tbs1.dbf' SIZE 70M
  SMALLFILE UNDO TABLESPACE sundo_tbs
  DATAFILE '/disk3/sundo_tbs1.dbf; SIZE 100M;

ACTIONS ON TABLESPACES

The first example shown below the tablespace name can be mentioned when resizing the datafile.

SQL> ALTER TABLESPACE USERS RESIZE 2G;

SQL> ALTER TABLESPACE USERS AUTOEXTEND ON;

CREATING A BIGFILE TABLESPACE

SYNTAX:

CREATE BIGFILE TABLESPACE ***tablespacename***
DATAFILE '*location of datafile*' SIZE n
[AUTOEXTEND clause]

The following example creates a bigfile tablespace bigtbs1 with a datafile bigtbs1.dbf of 200 MB:

SQL> CREATE BIGFILE TABLESPACE bigtbs1
DATAFILE 'bigtbs1.dbf'
SIZE 200MB
AUTOEXTEND ON;

DATA DICTIONARY CHANGES

- **DATABASE_PROPERTIES** data dictionary view: A new row is added to specify the default tablespace type for the database: BIGFILE or SMALLFILE.

```
SQL> SELECT property_value
       FROM database_properties
       WHERE property_name = 'DEFAULT_TBS_TYPE';
```

DBA_TABLESPACE – New BIGFILE that indicates if a tablespace is a bigfile tablespace or not.\
V$TABLESPACE : New BIGFILE that indicates if a tablespace is a bigfile tablespace or not.

CREATE TEMPORARY TABLESPACE GROUPS

In Oracle 10g, you can group temporary tablespaces into groups known as the TEMPORARY TABLESPACE GROUPS. It is a shortcut to a list of temporary tablespaces.

GUIDELINES

- A temporary tablespace group must contain at least one TEMPORARY tablespace.
- A permanent or temporary tablespace cannot have the same name as a temporary tablespace group.
- A temporary tablespace group name can appear any place a TEMPORARY tablespace name can appear. For e.g. when assigning a temporary tablespace to a user.
- The temporary tablespace group is not explicitly created. It is created implicitly when the first TEMPORARY tablespace is assigned to it, and deleted when the last TEMPORARY tablespace is removed from the group.
- A specific user can use multiple temporary tablespaces in different sessions at the same time for reasons of distributing workload.
- During a parallel execution, the slave processes can use multiple temporary tablespaces.
- Multiple default temporary tablespaces can be specified at the database level.

CREATING A TEMPORARY TABLESPACE GROUP

You can create temporary tablespace groups from EM console or using commands.

Using SQL command to create a temporary tablespace:

```
CREATE TEMPORARY TABLESPACE tablespacename
TEMPFILE ' location of file' SIZE n
[TABLESPACE GROUP groupname | ' '];
```

- In this first example you create a tablespace group by the name GROUP1 that contains a temporary tablespace called TEMP1

```
SQL> CREATE TEMPORARY TABLESPACE temp1
        TEMPFILE '/disk1/temp1.dbf'
        SIZE 100M
        TABLESPACE GROUP group1;
```

- In this second example we create a temporary tablespace called TEMP2. It does not belong to any group. This statement is equivalent to not specifying the tablespace group clause at all.

```
        SQL> CREATE TEMPORARY TABLESPACE temp2
        TEMPFILE '/disk2/temp2.dbf'
        SIZE 50M
        TABLESPACE GROUP ' ';
```

ADDING A TEMPORARY TABLESPACE TO A TABLESPACE GROUP

- You can use the ALTER TABLESPACE command to assign a temporary tablespace to a group, as in the example below. If the group with name GROUP2 does not exist, it will be created.

```
SQL> ALTER TABLESPACE temp2 TABLESPACE GROUP group2;
```

- To remove a tablespace from a group you can use:

```
SQL> ALTER TABLESPACE temp3 TABLESPACE GROUP ' ';
```

- To set a temporary tablespace group as the default for a database, you can issue:

```
SQL> ALTER DATABASE
        DEFAULT TEMPORARY TABLESPACE group1;
```

EXAM OBJECTIVE: ASSIGN TEMPORARY TABLESPACE GROUPS TO USERS

- You can assign a temporary tablespace group to a user when you create a user or after user creation using the ALTER user command.

```
SQL>CREATE USER userA
      IDENTIFIED BY userA
      DEFAULT TABLESPACE ts1
      TEMPORARY TABLESPACE group1;
```

The **DBA_TABLESPACE_GROUPS** lists all tablespaces contained in each temporary tablespace group.

```
SQL> SELECT group_name, tablespace_name
      FROM DBA_TABLESPACE_GROUPS;
```

EXAM OBJECTIVE: SKIP UNUSEABLE INDEXES

Prior to Oracle 10g, SKIP_UNUSABLE_INDEXES could be specified only at a session Level and was set with a default value of FALSE, and was used to prevent the Optimizer from using unusable indexes.

In Oracle 10g, the SKIP_UNUSABLE_INDEXES parameter can be specified at both an instance and at a session level, with a default value of TRUE.

By setting the value to TRUE, the optimizer can select an execution plan that does not utilize unusable indexes. This change prevents the ORA-01502 error.

EXAM OBJECTIVE: CREATE HASH-PARTITIONED GLOBAL INDEXES

In previous releases of Oracle, only range-partitioned global indexes were supported. In a range-partitioned global index, each index partition contains values defined by a partition bound.

Oracle 10g, introduces a new hash-partitioning method for global indexes. In a hash-partitioned global index, each partition contains values determined by a proprietary hash function based on the partitioning key and the number of partitions.

This new partitioning technique is useful in reducing index contention when a small number of leaf blocks in the index are frequently accessed in a multi-user online transaction processing (OLTP) environment.

With hash-partitioned global indexes, index entries are hashed to different partitions based on the partitioning key and the number of partitions, thus spreading contention over the number of partitions defined resulting in spreading the hot spots as there are partitions.

EXAM OBJECTIVE: CREATING HASH-PARTITIONED GLOBAL INDEXES

The CREATE INDEX commands is used to create hash-partitioned global indexes. The two types are displayed below:

1. Specify the description of each hash partition individually as displayed below:

```
SQL> CREATE INDEX global_hash_part_indx
     ON  line_items(orderno) GLOBAL
     PARTITION BY HASH(orderno)
     (       PARTITION p1 TABLESPACE tbs1,
             PARTITION p2 TABLESPACE tbs2,
             PARTITION p3 TABLESPACE tbs3,
             PARTITION p4 TABLESPACE tbs4
     );
```

EXAM OBJECTIVE: Maintain hash-partitioned global indexes

The maintenance commands supported for a hash-partitioned global index are:

1) Rebuilding of hash-partitioned indexes must be done partition-wise.
   ALTER INDEX … REBUILD PARTITION
   Example:
   SQL> ALTER INDEX npr REBUILD PARTITION P2;
2) Modifying the storage parameters, can be done using
   ALTER INDEX …. <modify_index_storage>
3) To modify the default attributes
   ALTER INDEX … MODIFY DEFAULT ATTRIBUTES
4) Modifying a partition, the UNUSABLE option is the only one supported.
   ALTER INDEX MODIFY PARTITION
5) Renaming a partition
   ALTER INDEX … RENAME PARTITION
6) Dropping the hash partitioned global index

DROP INDEX indexname;