Oracle Tablespace

Summary: in this tutorial, you will learn about the Oracle tablespace and how Oracle uses tablespaces to logically store the data in the database.

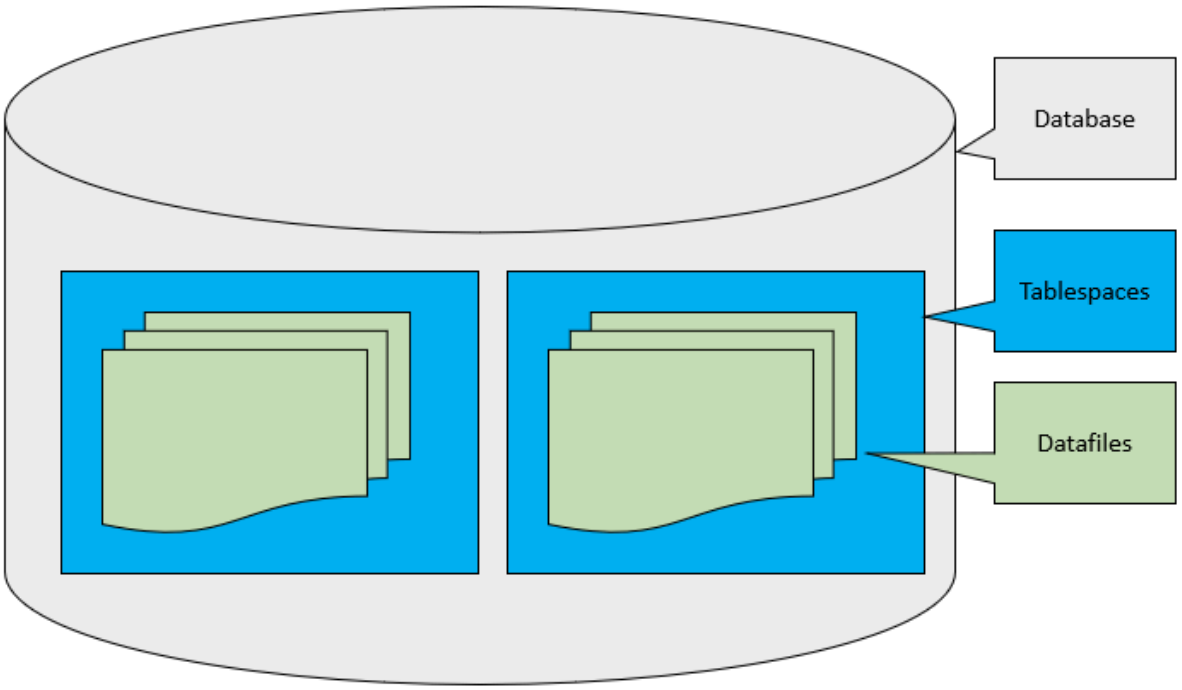
What is an Oracle Tablespace

Oracle divides a database into one or more logical storage units called tablespaces.

Each tablespace consists of one or more files called **datafiles**. A datafile physically stores the data objects of the database such as tables and indexes on disk.

In other words, Oracle **logically** stores data in the tablespaces and **physically** stores data in datafiles associated with the corresponding tablespaces.

The following picture illustrates the relationship between a database, tablespaces, and datafiles:



By using tablespaces, you can perform the following operations:

- Control the storage size allocated for the database data.

- Grant specific space quotas to the database users.
- Control the availability of data by taking tablespaces online or offline (more on this later).
- Improve the performance of the database by allocating data storage across devices.
- Perform partial database backup or recovery.

Default tablespaces in Oracle

Oracle comes with the following default tablespaces: `SYSTEM`, `SYSAUX`, `USERS`, `UNDOTBS1`, and `TEMP`.

- The `SYSTEM` and `SYSAUX` tablespaces store system-generated objects such as data dictionary tables. You should not store any object in these tablespaces.
- The `USERS` tablespace is helpful for ad-hoc users.
- The `UNDOTBS1` holds the undo data.
- The `TEMP` is the temporary tablespace that is used for storing intermediate results of sorting, hashing, and large object processing operations.

Online and Offline Tablespaces

A tablespace can be **online** or **offline**. If a tablespace is offline, you cannot access data stored in it. On the other hand, if a tablespace is online, its data is available for reading and writing.

> Note that the `SYSTEM` tablespace must always be online because it contains the data dictionary that must be available to Oracle.

Normally, a tablespace is online so that its data is available to users. However, you can take a tablespace offline to make data inaccessible to users when you update and maintain the applications.

If some errors occur such as hardware failures, Oracle automatically takes an online tablespace offline. If you attempt to access data in offline tablespace, you'll encounter an error.

Read-Only Tablespaces

The read-only tablespaces allow Oracle to avoid performing backup and recovery of large, static parts of a database.

Since Oracle doesn't update the files of a read-only tablespace, you can store the files on the read-only media.

Oracle allows you to remove objects such as tables and indexes from a read-only tablespace. However, it does not allow you to create or alter objects in a read-only tablespace.

When you create a new tablespace, it is in the read-write mode. To change a tablespace to a read-only tablespace, you use the `ALTER TABLESPACE` command with the `READ ONLY` option.

Bigfile and smallfile tablespaces

A smallfile table space can have **multiple** datafiles. Each datafile has a size limit. Typically, each has 32 GB, depending on block size and OS. The smallfile tablespaces is suitable for most applications.

Unlike a smallfile tablespace, a bigfile tablespace contains **only one** datafile. That single data file can be up to 128TB with 32KB block size. Typically, you use the bigfile tablespaces to simplify the storage management in very large databases (VLDBs) in Oracle Cloud.

Before Oracle 23ai, ,when you create a tablespace, it defaults to a smallfile tablespace.

However, starting from Oracle 23ai, a new tablespace defaults to a bigfile tablespace.

The following table displays the key differences between smallfile and bigfile tablespaces:

Feature	**Smallfile Tablespace**	**Bigfile Tablespace**

Max files per tablespace	1022	1
Max size per datafile	~32 GB	128 TB
Total size per tablespace	Up to 32 GB × 1022	Up to 128 TB
Suitable for	Most workloads	VLDBs, ASM, Oracle Cloud
Default	Yes, before Oracle 23ai.	Yes, in Oracle 23ai or later.

To check tablespace type, you can use the following query:

```
SELECT
  tablespace_name,
  bigfile
FROM
  dba_tablespaces;
```

Output:

TABLESPACE_NAME	BIGFILE
SYSTEM	YES
SYSAUX	YES
UNDOTBS1	YES
TEMP	NO
USERS	NO

Oracle CREATE TABLESPACE

Summary: In this tutorial, you will learn how to use the Oracle `CREATE TABLESPACE` statement to create a new tablespace in a database.

Introduction to the CREATE TABLESPACE statement

The `CREATE TABLESPACE` statement allows you to create a new tablespace.

The following statement shows how to create a new tablespace called`tbs1` with the size of 1MB:

```
CREATE TABLESPACE tbs
   DATAFILE 'tbs_data.dbf'
   SIZE 10m;
```
Code language: SQL (Structured Query Language) (sql)

In this statement:

- First, specify the name of the tablespace after the `CREATE TABLESPACE` keywords. In this example, the tablespace name is `tbs1`.
- Second, specify the path to the data file of the tablespace in the `DATAFILE` clause. In this case, it is `tbs1.dbf`. Note that you can use the datafile full path.
- Third, specify the size of the tablespace in the `SIZE` clause. In this example, `10m` stands for `10MB`, which is relatively small.

On Oracle 23ai or later, when you create a new tablespace, it defaults to a bigfile tablespace. If you want to create a smallfile tablespace, you need to use the `SMALLFILE` keyword:

```
CREATE TABLESPACE tbs_small
   DATAFILE 'tbs_small_data.dbf'
   SIZE 10m;
```

Once the tablespace is created, you can find its information by querying data from the `dba_data_files` view:

```
SELECT
   tablespace_name,
   file_name,
   bytes / 1024/ 1024  MB
FROM
   dba_data_files;
```
Code language: SQL (Structured Query Language) (sql)

Here are all the tablespaces in the current database:

TABLESPACE_NAME	FILE_NAME	MB
SYSTEM	/opt/oracle/oradata/FREE/FREEPDB1/system01.dbf	430
SYSAUX	/opt/oracle/oradata/FREE/FREEPDB1/sysaux01.dbf	920
UNDOTBS1	/opt/oracle/oradata/FREE/FREEPDB1/undotbs01.dbf	195
USERS	/opt/oracle/oradata/FREE/FREEPDB1/users01.dbf	300
TBS	/opt/oracle/product/23ai/dbhomeFree/dbs/tbs_data.dbf	10
TBS_SMALL	/opt/oracle/product/23ai/dbhomeFree/dbs/tbs_small_data.dbf	10

The `CREATE TABLESPACE` is quite complex with many options, you can find more information from the Oracle `CREATE TABLESPACE` page.

Tablespaces and the CREATE TABLE statement

When you create a new table, Oracle automatically places the table in the default tablespace of the user who created the table.

However, you can explicitly specify the tablespace to which the table belongs as shown in the following query:

```
CREATE TABLE table_name(
   ...
)
TABLESPACE tablespace_name;
```
Code language: SQL (Structured Query Language) (sql)

Note that you must have privileges on the tablespace that you specify in the `CREATE TABLE` statement. For example:

First, create a new table called `t1` whose tablespace is `tbs1`:

```
CREATE TABLE t1(
   id INT GENERATED ALWAYS AS IDENTITY,
   c1 VARCHAR2(32)
) TABLESPACE tbs;
```

Code language: SQL (Structured Query Language) (sql)

Second, insert 10,000 rows into the `t1` table:

```
BEGIN
   FOR counter IN 1..10000 loop
      INSERT INTO t1(c1)
      VALUES(sys_guid());
   END loop;
END;
/
```

Code language: SQL (Structured Query Language) (sql)

Third, check the free space of the `tbs1` tablespace by querying from the `dba_free_space` view:

```
SELECT
   tablespace_name,
   bytes / 1024 / 1024 MB
FROM
   dba_free_space
WHERE
   tablespace_name = 'TBS';
```

Code language: SQL (Structured Query Language) (sql)

Output:

```
TABLESPACE_NAME                          MB
------------------------------ ----------
TBS1                                0.9375
```

Code language: CSS (css)

Fourth, insert 100,000 rows into the `t1` table, Oracle will issue an error due to insufficient storage in the tablespace:

```
BEGIN
   FOR counter IN 1..100000 loop
      INSERT INTO t1(c1)
      VALUES(sys_guid());
   END loop;
END;
/
```

Code language: SQL (Structured Query Language) (sql)

Here is the error message:

```
ORA-01653: unable to increase tablespace TBS by 1MB
during insert or update on table OT.T1
```

Code language: SQL (Structured Query Language) (sql)

To fix this, you can resize the tablespace using the following `ALTER DATABASE` statement:

```
ALTER DATABASE
   DATAFILE 'tbs_data.dbf'
   RESIZE 10m;
```

Code language: SQL (Structured Query Language) (sql)

If you reinsert 100,000 rows into the t1 table, it should work.

The second way to avoid this issue, when creating a new tablespace, you can use the `AUTOEXTEND ON` clause as follows:

```
CREATE TABLESPACE tbs1
   DATAFILE 'tbs_data.dbf'
   SIZE 1m
   AUTOEXTEND 200m;
```

Code language: SQL (Structured Query Language) (sql)

Summary

- Use the Oracle `CREATE TABLESPACE` statement to create a new tablespace.

Oracle DROP TABLESPACE

Summary: in this tutorial, you will learn how to remove a tablespace from the database by using the Oracle `DROP TABLESPACE` statement.

Introduction to Oracle DROP TABLESPACE statement

The `DROP TABLESPACE` allows you to remove a tablespace from the database.

Here's the basic syntax of the `DROP TABLESPACE` statement:

```
DROP TABLESPACE tablespace_name
    [INCLUDING CONTENTS [AND | KEEP] DATAFILES]
    [CASCADE CONSTRAINTS];
```
Code language: SQL (Structured Query Language) (sql)

In this syntax:

- First, specify the name of the tablespace that you want to drop after the `DROP TABLESPACE` keywords.
- Second, use the `INCLUDE CONTENTS` to delete all contents of the tablespace. If the tablespace has any objects, you must use this option to remove the tablespace. Any attempt to remove a tablespace that has objects without specifying the `INCLUDING CONTENTS` option will result in an error.
- Third, use `AND DATAFILES` option to instruct Oracle to delete the datafiles of the tablespace and `KEEP DATAFILES` option to leave the datafiles untouched.
- Fourth, if the tablespace has objects such as tables whose primary keys are referenced by referential integrity constraints from tables outside the tablespace, you must use the `CASCADE CONSTRAINTS` option to drop these constraints. If you omit the `CASCACDE CONSTRAINTS` clause in such situations, Oracle returns an error and does not remove the tablespace.

You can use the `DROP TABLESPACE` to remove a tablespace regardless of whether it is online or offline. However, it's a good practice to take the tablespace offline before removing it to ensure that no sessions are currently accessing any objects in the tablespace.

> Note that you cannot drop the `SYSTEM` tablespace and only can drop the `SYSAUX` tablespace when you start the database in the `MIGRATE` mode.

You need to have the `DROP TABLESPACE` system privilege to execute the `DROP TABLESPACE` statement. To drop the `SYSAUX` tablespace, you need to have the `SYSDBA` system privilege.

Oracle DROP TABLESPACE statement examples

Let's take some examples of using the `DROP TABLESPACE` statement.

Using Oracle DROP TABLESPACE to remove an empty tablespace example

First, create a new tablespace named `tbs1`:

```
CREATE TABLESPACE tbs_empty
    DATAFILE 'tbs_empty_data.dbf'
    SIZE 10m;
```
Code language: SQL (Structured Query Language) (sql)

Second, use the `DROP TABLESPACE` to remove the `tbs1` tablespace:

```
DROP TABLESPACE tbs_empty;
```
Code language: SQL (Structured Query Language) (sql)

Using Oracle DROP TABLESPACE to remove a non-empty tablespace example

First, create a new tablespace named `tbs2`:

```
CREATE TABLESPACE tbs_non_empty
    DATAFILE 'tbs_non_empty.dbf'
    SIZE 50m;
```
Code language: SQL (Structured Query Language) (sql)

Second, create a new table `t2` in the tablespace `tbs2`:

```
CREATE TABLE t2 (
    c1 INT
) TABLESPACE tbs_non_empty;
```
Code language: SQL (Structured Query Language) (sql)

Third, use the `DROP TABLESPACE` statement to drop the `tbs2` tablespace:

```
DROP TABLESPACE tbs_non_empty;
```
Code language: SQL (Structured Query Language) (sql)

Oracle issued the following error:

```
ORA-01549: tablespace not empty, use `INCLUDING
CONTENTS` option
```
Code language: SQL (Structured Query Language) (sql)

To drop the `tbs_non_empty` tablespace, you need to use the `INCLUDING CONTENTS` option:

```
DROP TABLESPACE tbs_non_empty
    INCLUDING CONTENTS;
```
Code language: SQL (Structured Query Language) (sql)

Oracle issued the following message indicating that the tablespace has been dropped:

```
Tablespace dropped.
```
Code language: SQL (Structured Query Language) (sql)

Using Oracle DROP TABLESPACE to remove a tablespace whose tables are referenced by referential constraints

First, create two tablespaces named `tbs3` and `tbs4`:

```
CREATE SMALLFILE TABLESPACE tbs3
    DATAFILE 'tbs3_data.dbf'
    SIZE 5m;
```

```
CREATE SMALLFILE TABLESPACE  tbs4
    DATAFILE 'tbs4_data.dbf'
    SIZE 5m;
```
Code language: SQL (Structured Query Language) (sql)

Next, create a new table in the `tbs3` tablespace:

```
CREATE TABLE t3(
    c1 INT PRIMARY KEY
) TABLESPACE tbs3;
```
Code language: SQL (Structured Query Language) (sql)

Then, create a new table in the `tbs4` tablespace:

```
CREATE TABLE t4(
    c1 INT PRIMARY KEY,
    c2 INT NOT NULL,
    FOREIGN KEY(c2) REFERENCES t3(c1)
) TABLESPACE tbs4;
```
Code language: SQL (Structured Query Language) (sql)

After that, drop the tablespace `tbs3`:

```
DROP TABLESPACE tbs3
    INCLUDING CONTENTS;
```
Code language: SQL (Structured Query Language) (sql)

Oracle issued the following error:

```
ORA-02449: unique/primary keys in table referenced by
foreign keys
```
Code language: SQL (Structured Query Language) (sql)

Finally, use the `DROP TABLESPACE` that includes the `CASCADE CONSTRAINTS` option to drop the tablespace:

```
DROP TABLESPACE tbs3
INCLUDING CONTENTS AND DATAFILES
CASCADE CONSTRAINTS;
```

Code language: SQL (Structured Query Language) (sql)

It worked as expected.

Summary

- Use the Oracle `DROP TABLESPACE` statement to remove a tablespace from the database

Oracle Extend Tablespace

Summary: In this tutorial, you will learn how to extend the size of a tablespace in the Oracle Database.

When the tablespaces of the database are full, you will no longer be able to add or remove data from these tablespaces.

There are several ways to extend a tablespace.

Extending a tablespace by adding a new datafile

If you use a smallfile tablespace, you can use `ALTER TABLESPACE` statement to add a more datafile to the tablespace:

```
ALTER TABLESPACE tablespace_name
    ADD DATAFILE 'path_to_datafile'
    SIZE size;
```
Code language: SQL (Structured Query Language) (sql)

For both smallfile and bigfile tablespaces, you can use the `AUTOEXTEND ON` clause. Oracle will automatically extend the size of the datafile when needed:

```
ALTER TABLESPACE tablespace_name
    ADD DATAFILE 'path_to_datafile'
    SIZE size    AUTOEXTEND ON;
```
Code language: SQL (Structured Query Language) (sql)

Let's see the following example.

First, create a new tablespace called `tbs10` with the size of 10MB:

```
CREATE SMALLFILE TABLESPACE tbs10
    DATAFILE 'tbs10.dbf' SIZE 10m;
```
Code language: SQL (Structured Query Language) (sql)

Next, create a new table `t1` whose tablespace is `tbs10`:

```
CREATE TABLE t10(id INT PRIMARY  KEY)
TABLESPACE tbs10;
```
Code language: SQL (Structured Query Language) (sql)

Then, insert `1,000,000` rows into the `t1` table:

```
BEGIN
    FOR counter IN 1..1000000 loop
        INSERT INTO t10(id)
        VALUES(counter);
    END loop;
END;
/
```
Code language: SQL (Structured Query Language) (sql)

Oracle issued the following error:

```
ORA-01653: unable to increase tablespace TBS10 by 1MB
during insert or update on table OT.T1
```
Code language: SQL (Structured Query Language) (sql)

So the tablespace `tbs10` does not have enough space for the 1 million rows.

After that, use the `ATLER TABLESPACE` statement to add one more datafile whose size is 10MB with the `AUTOEXTEND ON` option:

```
ALTER TABLESPACE tbs10
    ADD DATAFILE 'tbs10_2.dbf'
    SIZE 10m
    AUTOEXTEND ON;
```
Code language: SQL (Structured Query Language) (sql)

Finally, insert 1 million rows into the `t1` table. It should work now. This query returns the number of rows from the `t1` table:

```
BEGIN
    FOR counter IN 1..1000000 loop
```

```
        INSERT INTO t10(id)
        VALUES(counter);
    END loop;
END;
/
```

You can retrieve the number of rows from the `t1` table:

```
SELECT count(*) FROM t1;
```
Code language: SQL (Structured Query Language) (sql)

Here is the output:

```
COUNT(*)
----------
1000000
```
Code language: SQL (Structured Query Language) (sql)

Extending a tablespace by resizing the datafile

You can use the `ALTER DATABASE RESIZE DATAFILE` statement to extend the size for smallfile and bigfile tablespaces:

```
ALTER DATABASE
    DATAFILE 'path_to_datafile'
    RESIZE size;
```
Code language: SQL (Structured Query Language) (sql)

For example:

First, create a new tablespace called `tbs11`:

```
CREATE TABLESPACE tbs11
    DATAFILE 'tbs11.dbf'
    SIZE 10m;
```
Code language: SQL (Structured Query Language) (sql)

Next, create a new table called `t2` that uses `tbs11` as the tablespace:

```
CREATE TABLE t11(
```

```
    c INT PRIMARY KEY
) TABLESPACE tbs11;
```
Code language: SQL (Structured Query Language) (sql)

Then, query the size of the tablespace `tbs11`:

```
SELECT
    tablespace_name,
    SUM(bytes)/1024/1024 AS total_mb
FROM
    dba_data_files
WHERE
    tablespace_name = 'TBS11'
GROUP BY tablespace_name;
```
Code language: SQL (Structured Query Language) (sql)

Output:

```
TABLESPACE_NAME                TOTAL_MB
------------------------------ ----------
TBS11                                  10
```
Code language: SQL (Structured Query Language) (sql)

After that, use the `ALTER DATABASE` to extend the size of the datafile of the tablespace to 15MB:

```
ALTER DATABASE
    DATAFILE 'tbs11.dbf'
    RESIZE 15m;
```
Code language: SQL (Structured Query Language) (sql)

Finally, query the total size of the `tbs11` tablespace:

```
SELECT
    tablespace_name,
    SUM(bytes)/1024/1024 AS total_mb
FROM
    dba_data_files
```

```
WHERE
    tablespace_name = 'TBS11'
GROUP BY tablespace_name;
```
Code language: SQL (Structured Query Language) (sql)

Here is the output:

```
TABLESPACE_NAME                   TOTAL_MB
------------------------------ ----------
TBS11                                  15
```
Code language: SQL (Structured Query Language) (sql)

As you can see, the size of the tablespace `tbs11` has been extended to 15 MB.

Note that Oracle does not allow you to add a datafile to a bigfile tablespace, therefore, you can only use `ALTER DATABASE DATAFILE RESIZE` command.

Summary

- Use `ALTER TABLESPACE ... ADD DATA FILE` to extend a smallfile tablespace.
- Use `ATLER DATABASE ... RESIZE` to extend a smallfile or bigfile tablesppaces.

Oracle Tablespace Group

Summary: In this tutorial, you will learn about Oracle temporary tablespace groups and how to use the tablespace group effectively to optimize internal Oracle operations.

Introduction to Oracle tablespace groups

A **tablespace group** is a logical grouping of one or more temporary tablespaces.

A tablespace group enables Oracle to **distribute temporary data** across multiple temporary tablespaces, providing **better performance and scalability**.

The following describes the property of a tablespace group:

- A tablespace group must contain at least one temporary tablespace.
- The name of a tablespace group cannot be the same as any tablespace.
- A tablespace group can be assigned as a *default temporary tablespace* for the database or a temporary tablespace for a user.

Creating a tablespace group

Oracle does not provide a statement to create a tablespace group. However, you can create a tablespace group when you assign the first temporary tablespace to the group using the `CREATE TEMPORARY TABLESPACE` statement:

```
CREATE TEMPORARY TABLESPACE tablespace_name
    TEMPFILE 'path_to_file'
    SIZE 50M
    TABLESPACE GROUP group_name;
```

Code language: SQL (Structured Query Language) (sql)

Or `ALTER TABLESPACE` statement:

```
ALTER TABLESPACE tablespace_name
```

```
    TABLESPACE GROUP group_name;
```
Code language: SQL (Structured Query Language) (sql)

Removing a tablespace group

Oracle automatically drops a tablespace group when you remove the last temporary tablespace from the tablespace group.

Viewing tablespace groups

The view `DBA_TABLESPACE_GROUPS` lists all tablespace groups and their member temporary tablespace.

```
SELECT
  tablespace_name,
  group_name
FROM
  dba_tablespace_groups;
```
Code language: SQL (Structured Query Language) (sql)

Moving a tablespace to another tablespace group

To move a temporary tablespace to another tablespace group, you use the `ALTER TABLESPACE` statement.

The following statement moves the temporary tablespace `tablespace_name` to the tablespace group `destination`:

```
ALTER TABLESPACE tablespace_name
TABLESPACE GROUP destination;
```
Code language: SQL (Structured Query Language) (sql)

Note that the `destination` tablespace group must exist.

Assigning a tablespace group as the default temporary tablespace

To assign a tablespace group as the default temporary tablespace for a database, you use the `ALTER DATABASE DEFAULT TEMPORARY TABLESPACE` statement:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE
tablespace_group;
```
Code language: SQL (Structured Query Language) (sql)

All users who have not been assigned a temporary tablespace will use the temporary tablespaces contained in the `tablespace_group`.

Note that you cannot drop any temporary tablespace that belongs to a tablespace group that is specified as a default temporary tablespace. In this case, you first need to de-assign the temporary tablespace from the tablespace group and then remove it.

Benefits of using a tablespace group

A tablespace group brings the following benefits:

- Enable multiple default temporary tablespaces to be used at the database level.
- Allow the user to use multiple temporary tablespaces simultaneously across different sessions.
- Reduce the contention when you have multiple temporary tablespaces.

Oracle tablespace group examples

First, create a new temporary tablespace and assign it to the tablespace group `tbs1`:

```
CREATE TEMPORARY TABLESPACE temp2
TEMPFILE 'temp2.dbf'
SIZE 100M
TABLESPACE GROUP tbsg1;
```
Code language: SQL (Structured Query Language) (sql)

Since the tablespace group `tbsg1` does not exist, the statement creates the tablespace group `tbsg1`.

Second, assign the temp temporary tablespace `temp` to the `tbsg1` tablespace group:

```
ALTER TABLESPACE temp TABLESPACE GROUP tbsg1;
```
Code language: SQL (Structured Query Language) (sql)

Third, assign the tablespace group gbsg1 as the default temporary tablespace:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tbsg1;
```
Code language: SQL (Structured Query Language) (sql)

Finally, verify the current default temporary tablespace:

```
SELECT
    property_name,
    property_value
FROM
    database_properties
WHERE
    property_name='DEFAULT_TEMP_TABLESPACE';
```
Code language: SQL (Structured Query Language) (sql)

PROPERTY_NAME	PROPERTY_VALUE
DEFAULT TEMP TABLESPACE	TBSG1

Summary

- A tablespace group is a logical grouping of one or more **temporary tablespaces**.
- Use tablespace groups to distribute temporary data across multiple temporary tablespaces for **better performance and scalability**.