How-To Use CREATE DATABASE Statement
Oracle SQL *Plus

Using the `CREATE DATBASE` SQL statement is a more manual approach to creating a database. One advantage of using this statement over using DBCA is that you can create databases from within scripts.

If you use the `CREATE DATABASE` statement, you must complete additional actions before you have an operational database. These actions include building views on the data dictionary tables and installing standard PL/SQL packages. You perform these actions by running the supplied scripts.

If you have existing scripts for creating your database, consider editing those scripts to take advantage of new Oracle Database features.

The instructions in this section apply to *single-instance installations only*. Refer to the Oracle Real Application Clusters (Oracle RAC) installation guide for your platform for instructions for creating an Oracle RAC database.

Note:

Single-instance does not mean that only one Oracle instance can reside on a single host computer. In fact, multiple Oracle instances (and their associated databases) can run on a single host computer. A **single-instance database** is a database that is accessed by only one Oracle instance, as opposed to an Oracle RAC database, which is accessed concurrently by multiple Oracle instances on multiple nodes.

Complete the following steps to create a database with the `CREATE DATABASE` statement. The examples create a database named `mynewdb`.

Step 1: Specify an Instance Identifier (SID)

Decide on a unique Oracle system identifier (SID) for your instance, open a command window, and set the `ORACLE_SID` environment variable. Use this command windows for the subsequent steps.

`ORACLE_SID` is used to distinguish this instance from other Oracle Database instances that you may create later and run concurrently on the same host computer. The maximum number of characters for `ORACLE_SID` is 12, and only letters and numeric digits are permitted. On some platforms, the SID is case-sensitive.

Note:
It is common practice to set the SID to be equal to the database name. The maximum number of characters for the database name is eight.

The following example for UNIX and Linux operating systems sets the SID for the instance that you will connect to in Step 6: Connect to the Instance:

- Bourne, Bash, or Korn shell:

```
ORACLE_SID=mynewdb
export ORACLE_SID
```

- C shell:

```
setenv ORACLE_SID mynewdb
```

The following example sets the SID for the Windows operating system:

```
set ORACLE_SID=mynewdb
```

Step 2: Ensure That the Required Environment Variables Are Set

Depending on your platform, before you can start SQL*Plus (as required in Step 6: Connect to the Instance), you may have to set environment variables, or at least verify that they are set properly.

For example, on most platforms, `ORACLE_SID` and `ORACLE_HOME` must be set. In addition, it is advisable to set the `PATH` variable to include the *ORACLE_HOME*/bin directory. On the UNIX and Linux platforms, you must set these environment variables manually. On the Windows platform, OUI automatically assigns values to `ORACLE_HOME` and `ORACLE_SID` in the Windows registry. If you did not create a database upon installation, OUI does not set `ORACLE_SID` in the registry, and you will have to set the `ORACLE_SID` environment variable when you create your database later.

Step 3: Choose a Database Administrator Authentication Method

You must be authenticated and granted appropriate system privileges in order to create a database. You can be authenticated as an administrator with the required privileges in the following ways:

- With a password file
- With operating system authentication

In this step, you decide on an authentication method.

To be authenticated with a password file, create the password file as described in "Creating and Maintaining a Password File". To be authenticated with operating system authentication, ensure that you log in to the host computer with a user account that is a member of the appropriate operating system user group. On the UNIX and Linux platforms, for example, this is typically the `dba` user group. On the Windows platform, the user installing the Oracle software is automatically placed in the required user group.

Step 4: Create the Initialization Parameter File

When an Oracle instance starts, it reads an initialization parameter file. This file can be a text file, which can be created and modified with a text editor, or a binary file, which is created and dynamically modified by the database. The binary file, which is preferred, is called a **server parameter file**. In this step, you create a text initialization parameter file. In a later step, you create a server parameter file from the text file.

One way to create the text initialization parameter file is to edit the sample presented in "Sample Initialization Parameter File".

If you create the initialization parameter file manually, ensure that it contains at least the parameters listed in Table 2-2. All other parameters not listed have default values.

Table 2-2 Recommended Minimum Initialization Parameters

Parameter Name	Mandatory	Notes
`DB_NAME`	Yes	Database identifier. Must correspond to the value used in the `CREATE DATABASE` statement. Maximum 8 characters.
`CONTROL_FILES`	No	Strongly recommended. If not provided, the database instance creates one control file in the same location as the initialization parameter file. Providing this parameter enables you to multiplex control files. See "Creating Initial Control Files" for more information.
`MEMORY_TARGET`	No	Sets the total amount of memory used by the instance and enables automatic memory management. You can choose other initialization parameters instead of this one for more manual control of memory usage. See "Configuring Memory Manually".

For convenience, store your initialization parameter file in the Oracle Database default location, using the default file name. Then when you start your database, it will not be necessary to specify the `PFILE` clause of the `STARTUP` command, because Oracle Database automatically looks in the default location for the initialization parameter file.

For more information about initialization parameters and the initialization parameter file, including the default name and location of the initialization parameter file for your platform, see "About Initialization Parameters and Initialization Parameter Files".

Step 5: (Windows Only) Create an Instance

On the Windows platform, before you can connect to an instance, you must manually create it if it does not already exist. The ORADIM command creates an Oracle instance by creating a new Windows service.

To create an instance:

- Enter the following command at a Windows command prompt:

  ```
  oradim -NEW -SID sid -STARTMODE MANUAL -PFILE pfile
  ```

  where *sid* is the desired SID (for example mynewdb) and *pfile* is the full path to the text initialization parameter file. This command creates the instance but does not start it.

> **Caution:**
> Do not set the -STARTMODE argument to AUTO at this point, because this causes the new instance to start and attempt to mount the database, which does not exist yet. You can change this parameter to AUTO, if desired, in Step 14.

See the section "Using ORADIM to Administer an Oracle Database Instance" in *Oracle Database Platform Guide for Microsoft Windows* for more information on the ORADIM command.

Step 6: Connect to the Instance

Start SQL*Plus and connect to your Oracle Database instance with the SYSDBA system privilege.

- To authenticate with a password file, enter the following commands, and then enter the SYS password when prompted:

  ```
  $ sqlplus /nolog
  SQL> CONNECT SYS AS SYSDBA
  ```

- To authenticate with operating system authentication, enter the following commands:

  ```
  $ sqlplus /nolog
  SQL> CONNECT / AS SYSDBA
  ```

SQL*Plus outputs the following message:

```
Connected to an idle instance.
```

Note:

SQL*Plus may output a message similar to the following:

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

If so, this means that the instance is already started. You may have connected to the wrong instance. Exit SQL*Plus with the `EXIT` command, check that `ORACLE_SID` is set properly, and repeat this step.

Step 7: Create a Server Parameter File

The server parameter file enables you to change initialization parameters with the `ALTER SYSTEM` command and persist the changes across a database shutdown and startup. You create the server parameter file from your edited text initialization file.

The following SQL*Plus command reads the text initialization parameter file (PFILE) with the default name from the default location, creates a server parameter file (SPFILE) from the text initialization parameter file, and writes the SPFILE to the default location with the default SPFILE name.

```
CREATE SPFILE FROM PFILE;
```

You can also supply the file name and path for both the PFILE and SPFILE if you are not using default names and locations.

Tip:
The database must be restarted before the server parameter file takes effect.

Note:
Although creating a server parameter file is optional at this point, it is recommended. If you do not create a server parameter file, the instance continues to read the text initialization parameter file whenever it starts.

Important—If you are using Oracle Managed Files and your initialization parameter file does not contain the `CONTROL_FILES` parameter, you must create a server parameter file now so the database can save the names and location of the control files that it creates during the `CREATE DATABASE` statement. See "Specifying Oracle Managed Files at Database Creation" for more information.

See Also:

- "Managing Initialization Parameters Using a Server Parameter File"

- *Oracle Database SQL Language Reference* for more information on the `CREATE SPFILE` command

Step 8: Start the Instance

Start an instance without mounting a database. Typically, you do this only during database creation or while performing maintenance on the database. Use the `STARTUP` command with the `NOMOUNT` clause. In this example, because the initialization parameter file or server parameter file is stored in the default location, you are not required to specify the `PFILE` clause:

```
STARTUP NOMOUNT
```

At this point, the instance memory is allocated and its processes are started. The database itself does not yet exist.

See Also:

- *Oracle Database Concepts* for an overview of the Oracle instance.
- "Managing Initialization Parameters Using a Server Parameter File"
- Chapter 3, "Starting Up and Shutting Down", to learn how to use the `STARTUP` command

Step 9: Issue the CREATE DATABASE Statement

To create the new database, use the `CREATE DATABASE` statement.

Example 1

The following statement creates database `mynewdb`. This database name must agree with the `DB_NAME` parameter in the initialization parameter file. This example assumes the following:

- The initialization parameter file specifies the number and location of control files with the `CONTROL_FILES` parameter.
- The directory `/u01/app/oracle/oradata/mynewdb` exists.
- The directories `/u01/logs/my` and `/u02/logs/my` exist.

```
CREATE DATABASE mynewdb
    USER SYS IDENTIFIED BY sys_password
    USER SYSTEM IDENTIFIED BY system_password
    LOGFILE GROUP 1 ('/u01/logs/my/redo01a.log','/u02/logs/my/redo01b.log') SIZE 100M BLOCKSIZE 512,
              GROUP 2 ('/u01/logs/my/redo02a.log','/u02/logs/my/redo02b.log') SIZE 100M BLOCKSIZE
512,
              GROUP 3 ('/u01/logs/my/redo03a.log','/u02/logs/my/redo03b.log') SIZE 100M BLOCKSIZE 512
```

```
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
CHARACTER SET US7ASCII
NATIONAL CHARACTER SET AL16UTF16
EXTENT MANAGEMENT LOCAL
DATAFILE '/u01/app/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
SYSAUX DATAFILE '/u01/app/oracle/oradata/mynewdb/sysaux01.dbf' SIZE 325M REUSE
DEFAULT TABLESPACE users
    DATAFILE '/u01/app/oracle/oradata/mynewdb/users01.dbf'
    SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
DEFAULT TEMPORARY TABLESPACE tempts1
    TEMPFILE '/u01/app/oracle/oradata/mynewdb/temp01.dbf'
    SIZE 20M REUSE
UNDO TABLESPACE undotbs
    DATAFILE '/u01/app/oracle/oradata/mynewdb/undotbs01.dbf'
    SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

A database is created with the following characteristics:

- The database is named `mynewdb`. Its global database name is `mynewdb.us.example.com`, where the domain portion (`us.example.com`) is taken from the initialization file. See "Determining the Global Database Name".
- Three control files are created as specified by the `CONTROL_FILES` initialization parameter, which was set before database creation in the initialization parameter file.

  See "Sample Initialization Parameter File" and "Specifying Control Files".
- The passwords for user accounts `SYS` and `SYSTEM` are set to the values that you specified. Beginning with Release 11*g*, the passwords are case-sensitive. The two clauses that specify the passwords for `SYS` and `SYSTEM` are not mandatory in this release of Oracle Database. However, if you specify either clause, you must specify both clauses. For further information about the use of these clauses, see "Protecting Your Database: Specifying Passwords for Users SYS and SYSTEM".
- The new database has three redo log file groups, each with two members, as specified in the `LOGFILE` clause. `MAXLOGFILES`, `MAXLOGMEMBERS`, and `MAXLOGHISTORY` define limits for the redo log. See "Choosing the Number of Redo Log Files". The block size for the redo logs is set to 512 bytes, the same size as physical sectors on disk. The `BLOCKSIZE` clause is optional if block size is to be the same as physical sector size (the default). Typical sector size and thus typical block size is 512. Permissible values for `BLOCKSIZE` are 512, 1024, and 4096. For newer disks with a 4K sector size, optionally specify `BLOCKSIZE` as 4096. See "Planning the Block Size of Redo Log Files" for more information.
- `MAXDATAFILES` specifies the maximum number of datafiles that can be open in the database. This number affects the initial sizing of the control file.

Note:
You can set several limits during database creation. Some of these limits are limited by and affected by

---

operating system limits. For example, if you set MAXDATAFILES, Oracle Database allocates enough space in the control file to store MAXDATAFILES filenames, even if the database has only one datafile initially. However, because the maximum control file size is limited and operating system dependent, you might not be able to set all CREATE DATABASE parameters at their theoretical maximums.

For more information about setting limits during database creation, see the *Oracle Database SQL Language Reference* and your operating system–specific Oracle documentation.

- The US7ASCII character set is used to store data in this database.

- The AL16UTF16 character set is specified as the NATIONAL CHARACTER SET, used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2.
- The SYSTEM tablespace, consisting of the operating system file /u01/app/oracle/oradata/mynewdb/system01.dbf is created as specified by the DATAFILE clause. If a file with that name already exists, it is overwritten.
- The SYSTEM tablespace is created as a locally managed tablespace. See "Creating a Locally Managed SYSTEM Tablespace".
- A SYSAUX tablespace is created, consisting of the operating system file /u01/app/oracle/oradata/mynewdb/sysaux01.dbf as specified in the SYSAUX DATAFILE clause. See "About the SYSAUX Tablespace".
- The DEFAULT TABLESPACE clause creates and names a default permanent tablespace for this database.
- The DEFAULT TEMPORARY TABLESPACE clause creates and names a default temporary tablespace for this database. See "Creating a Default Temporary Tablespace".
- The UNDO TABLESPACE clause creates and names an undo tablespace that is used to store undo data for this database if you have specified UNDO_MANAGEMENT=AUTO in the initialization parameter file. If you omit this parameter, it defaults to AUTO. See "Using Automatic Undo Management: Creating an Undo Tablespace".
- Redo log files will not initially be archived, because the ARCHIVELOG clause is not specified in this CREATE DATABASE statement. This is customary during database creation. You can later use an ALTER DATABASE statement to switch to ARCHIVELOG mode. The initialization parameters in the initialization parameter file for mynewdb relating to archiving are LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_FORMAT. See Chapter 13, "Managing Archived Redo Logs".

Tips:

- Ensure that all directories used in the `CREATE DATABASE` statement exist. The `CREATE DATABASE` statement does not create directories.
- If you are not using Oracle Managed Files, every tablespace clause must include a `DATAFILE` or `TEMPFILE` clause.
- If database creation fails, you can look at the alert log to determine the reason for the failure and to determine corrective actions. See "Viewing the Alert Log". If you receive an error message that contains a process number, examine the trace file for that process. Look for the trace file that contains the process number in the trace file name. See "Finding Trace Files" for more information.
- If you want to resubmit the `CREATE DATABASE` statement after a failure, you must first shut down the instance and delete any files created by the previous `CREATE DATABASE` statement.

Example 2

This example illustrates creating a database with Oracle Managed Files, which enables you to use a much simpler `CREATE DATABASE` statement. To use Oracle Managed Files, the initialization parameter `DB_CREATE_FILE_DEST` must be set. This parameter defines the base directory for the various database files that the database creates and automatically names. The following statement is an example of setting this parameter in the initialization parameter file:

```
DB_CREATE_FILE_DEST='/u01/app/oracle/oradata'
```

With Oracle Managed Files and the following `CREATE DATABASE` statement, the database creates the `SYSTEM` and `SYSAUX` tablespaces, creates the additional tablespaces specified in the statement, and chooses default sizes and properties for all datafiles, control files, and redo log files. Note that these properties and the other default database properties set by this method may not be suitable for your production environment, so it is recommended that you examine the resulting configuration and modify it if necessary.

```
CREATE DATABASE mynewdb
USER SYS IDENTIFIED BY sys_password
USER SYSTEM IDENTIFIED BY system_password
EXTENT MANAGEMENT LOCAL
DEFAULT TEMPORARY TABLESPACE temp
UNDO TABLESPACE undotbs1
DEFAULT TABLESPACE users;
```

Tip:

If your CREATE DATABASE statement fails, and if you did not complete Step 7, ensure that there is not a pre-existing server parameter file (SPFILE) for this instance that is setting initialization parameters in an unexpected way. For example, an SPFILE contains a setting for the complete path to all control files, and the CREATE DATABASE statement fails if those control files do not exist. Ensure that you shut down and restart the instance (with STARTUP NOMOUNT) after removing an unwanted SPFILE. See "*Managing Initialization Parameters Using a Server Parameter File*" for more information.

See Also:

- "Specifying CREATE DATABASE Statement Clauses"
- "Specifying Oracle Managed Files at Database Creation"
- Chapter 17, "Using Oracle Managed Files"
- *Oracle Database SQL Language Reference* for more information about specifying the clauses and parameter values for the CREATE DATABASE statement

Step 10: Create Additional Tablespaces

To make the database functional, you need to create additional tablespaces for your application data. The following sample script creates some additional tablespaces:

```
CREATE TABLESPACE apps_tbs LOGGING
      DATAFILE '/u01/app/oracle/oradata/mynewdb/apps01.dbf'
      SIZE 500M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
      EXTENT MANAGEMENT LOCAL;
-- create a tablespace for indexes, separate from user tablespace (optional)
CREATE TABLESPACE indx_tbs LOGGING
      DATAFILE '/u01/app/oracle/oradata/mynewdb/indx01.dbf'
      SIZE 100M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
      EXTENT MANAGEMENT LOCAL;
```

For information about creating tablespaces, see Chapter 14, "Managing Tablespaces".

Step 11: Run Scripts to Build Data Dictionary Views

Run the scripts necessary to build data dictionary views, synonyms, and PL/SQL packages, and to support proper functioning of SQL*Plus:

```
@?/rdbms/admin/catalog.sql
@?/rdbms/admin/catproc.sql
@?/sqlplus/admin/pupbld.sql
EXIT
```

The at-sign (@) is shorthand for the command that runs a SQL*Plus script. The question mark (?) is a SQL*Plus variable indicating the Oracle home directory. The following table contains descriptions of the scripts:

Script	Description
CATALOG.SQL	Creates the views of the data dictionary tables, the dynamic performance views, and public synonyms for many of the views. Grants PUBLIC access to the synonyms.
CATPROC.SQL	Runs all scripts required for or used with PL/SQL.
PUPBLD.SQL	Required for SQL*Plus. Enables SQL*Plus to disable commands by user.

Step 12: (Optional) Run Scripts to Install Additional Options

You may want to run other scripts. The scripts that you run are determined by the features and options you choose to use or install. Many of the scripts available to you are described in the *Oracle Database Reference.*

If you plan to install other Oracle products to work with this database, see the installation instructions for those products. Some products require you to create additional data dictionary tables. Usually, command files are provided to create and load these tables into the database data dictionary.

See your Oracle documentation for the specific products that you plan to install for installation and administration instructions.

Step 13: Back Up the Database.

Take a full backup of the database to ensure that you have a complete set of files from which to recover if a media failure occurs. For information on backing up a database, see *Oracle Database Backup and Recovery User's Guide.*

Step 14: (Optional) Enable Automatic Instance Startup

You might want to configure the Oracle instance to start automatically when its host computer restarts. See your operating system documentation for instructions. For example, on Windows, use the following command to configure the database service to start the instance upon computer restart:

```
ORADIM -EDIT -SID sid -STARTMODE AUTO -SRVCSTART SYSTEM [-SPFILE]
```

You must use the -SPFILE argument if you want the instance to read an SPFILE upon automatic restart.

Courtesy: https://docs.oracle.com/cd/E18283_01/server.112/e17120/create003.htm

Modified: 2021.10.09.11.18.AM

Dököll Solutions, Inc.