How to Create a User and Grant Permissions in Oracle

As always, begin by connecting to your server where Oracle is hosted, then connect to Oracle itself as the SYSTEM account.

The SYSTEM account is one of a handful of predefined administrative accounts generated automatically when Oracle is installed. SYSTEM is capable of most administrative tasks, but the task we’re particularly interested in is account management.

**Creating a User**

Once connected as SYSTEM, simply issue the [**CREATE USER**](https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_8003.htm) command to generate a new account.

CREATE USER books\_admin IDENTIFIED BY MyPassword;

Here we’re simply creating a books\_admin account that is IDENTIFIED or authenticated by the specified password.

## The Grant Statement

With our new books\_admin account created, we can now begin adding privileges to the account using the [**GRANT**](https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_9013.htm) statement. GRANT is a very powerful statement with many possible options, but the core functionality is to manage the privileges of both users and roles throughout the database.

## Providing Roles

Typically, you’ll first want to assign privileges to the user through attaching the account to various roles, starting with the CONNECT role:

GRANT CONNECT TO books\_admin;

In some cases to create a more powerful user, you may also consider adding the RESOURCE role (allowing the user to create named types for custom schemas) or even the DBA role, which allows the user to not only create custom named types but alter and destroy them as well.

GRANT CONNECT, RESOURCE, DBA TO books\_admin;

## Assigning Privileges

Next you’ll want to ensure the user has privileges to actually connect to the database and create a session using GRANT CREATE SESSION. We’ll also combine that with all privileges using GRANT ANY PRIVILEGES.

GRANT CREATE SESSION GRANT ANY PRIVILEGE TO books\_admin;

We also need to ensure our new user has disk space allocated in the system to actually create or modify tables and data, so we’ll GRANT TABLESPACE like so:

GRANT UNLIMITED TABLESPACE TO books\_admin;

## Table Privileges

While not typically necessary in newer versions of Oracle, some older installations may require that you manually specify the access rights the new user has to a specific schema and database tables.

For example, if we want our books\_admin user to have the ability to perform SELECT, UPDATE, INSERT, and DELETE capabilities on the books table, we might execute the following GRANT statement:

GRANT

SELECT,

INSERT,

UPDATE,

DELETE

ON

schema.books

TO

books\_admin;

This ensures that books\_admin can perform the four basic statements for the books table that is part of the schema schema.

There is a "grant all" option for tables. But before you reach for this, be aware that not only does it include the DML permissions above, it also gives:

* alter
* debug
* flashback
* index
* on commit refresh
* query rewrite
* read
* references

## Password Management

A brief note on password rules. By default the password will expire every 180 days. Which can lead to [ORA-28002 errors](https://blogs.oracle.com/sql/how-to-fix-ora-28002-the-password-will-expire-in-7-days-errors) on login.

Not only is this kinda annoying, it goes against [current password guidelines](https://www.troyhunt.com/passwords-evolved-authentication-guidance-for-the-modern-era/#donotmandateregularpasswordchanges). You can get around this by [changing the password\_life\_time](https://blogs.oracle.com/sql/how-to-fix-ora-28002-the-password-will-expire-in-7-days-errors" \l "database-user-profiles) for the user's profile.

While you're at it, you probably want to stop people picking short, easy to crack passwords. You can define a [password complexity function](http://www.oracle.com/pls/topic/lookup?ctx=dblatest&id=GUID-00BED08C-E064-4824-9697-C7219CB8D3AB) to do this.

So you’ve created your application user.

But you still need to assign it permissions on data\_owner’s objects. For table level access, you can give access to query and change the rows with:

Anyone with access to your network can connect as data\_owner. At which point they’re free to wreak havoc in your database.

This is a tricky problem to avoid. You can stop people getting in by locking the account with:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1 | alter user data\_owner account lock; |

## Removing Access

Over time applications get decommissioned. Or rewritten to access different information. But usually the data remains.

Leaving the user with access to unneeded data is a security risk. Stay on top of this and remove access when it’s no longer needed.

To do this, use the revoke command. This states what you’re removing from who. For system privileges this is:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1 | revoke create table from data\_owner; |

For object privileges, include the thing you're removing access from:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1 | revoke select on data\_owner.important\_stuff from app\_user; |

Remember: if your release scripts have grants for existing objects you'll need to undo these if you have to rollback. So ensure you include the corresponding revoke in your rollback scripts!

## Dropping Users

Getting rid of unwanted users is easy. Drop them with:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1 | drop user <username>; |

You can only do this if the user is not connected to the database. So ensure you clear up any sessions it has before you do so.

And there’s another step you need to watch for. Run this for data\_owner and you’re likely to hit this error:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1  2  3 | drop user data\_owner;    ORA-01922: CASCADE must be specified to drop 'DATA\_OWNER' |

Why?

You can’t remove users that own objects!

So you need to go in and drop all its tables, views, etc. Or do it in one shot with:

[?](https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database)

|  |  |
| --- | --- |
| 1 | drop user data\_owner cascade; |

This is an easy way to wipe all your data. So use with care!

CREATE TABLESPACE

**Purpose**

Use the CREATE TABLESPACE statement to create a **tablespace**, which is an allocation of space in the database that can contain schema objects.

* A **permanent tablespace** contains persistent schema objects. Objects in permanent tablespaces are stored in **datafiles**.
* An **undo tablespace** is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode. Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
* A **temporary tablespace** contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in **tempfiles**.

When you create a tablespace, it is initially a read/write tablespace. You can subsequently use the ALTER TABLESPACE statement to take the tablespace offline or online, add datafiles or tempfiles to it, or make it a read-only tablespace.

You can also drop a tablespace from the database with the DROP TABLESPACE statement.

## #1 - PERMANENT TABLESPACE

A permanent tablespace contains persistent schema objects that are stored in data files.

### Syntax

The syntax for the CREATE TABLESPACE statement when creating a permanent tablespace is:

CREATE

[ SMALLFILE | BIGFILE ]

TABLESPACE tablespace\_name

{ DATAFILE { [ 'filename' | 'ASM\_filename' ]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

[ AUTOEXTEND

{ OFF

| ON [ NEXT integer [ K | M | G | T | P | E ] ]

[ MAXSIZE { UNLIMITED | integer [ K | M | G | T | P | E ] } ]

}

]

| [ 'filename | ASM\_filename'

| ('filename | ASM\_filename'

[, 'filename | ASM\_filename' ] )

]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

}

{ MINIMUM EXTENT integer [ K | M | G | T | P | E ]

| BLOCKSIZE integer [ K ]

| { LOGGING | NOLOGGING }

| FORCE LOGGING

| DEFAULT [ { COMPRESS | NOCOMPRESS } ]

storage\_clause

| { ONLINE | OFFLINE }

| EXTENT MANAGEMENT

{ LOCAL

[ AUTOALLOCATE

| UNIFORM

[ SIZE integer [ K | M | G | T | P | E ] ]

]

| DICTIONARY

}

| SEGMENT SPACE MANAGEMENT { AUTO | MANUAL }

| FLASHBACK { ON | OFF }

[ MINIMUM EXTENT integer [ K | M | G | T | P | E ]

| BLOCKSIZE integer [ K ]

| { LOGGING | NOLOGGING }

| FORCE LOGGING

| DEFAULT [ { COMPRESS | NOCOMPRESS } ]

storage\_clause

| { ONLINE | OFFLINE }

| EXTENT MANAGEMENT

{ LOCAL

[ AUTOALLOCATE | UNIFORM [ SIZE integer [ K | M | G | T | P | E ] ] ]

| DICTIONARY

}

| SEGMENT SPACE MANAGEMENT { AUTO | MANUAL }

| FLASHBACK { ON | OFF }

]

}

**SMALLFILE**

A tablespace that contains 1,022 data or temp files (each file can be up to 4 million blocks in size). This is the most common tablespace size to create.

**BIGFILE**

A tablespace that contains only one data or temp file (this file can be up to 4 million blocks in size).

**TIP:** If you omit the SMALLFILE or BIGFILE option, the Oracle database will use the default tablespace type.

**tablespace\_name**

The name of the tablespace to create.

**storage\_clause**

The syntax for the storage\_clause is:

STORAGE

({ INITIAL integer [ K | M | G | T | P | E ]

| NEXT integer [ K | M | G | T | P | E ]

| MINEXTENTS integer

| MAXEXTENTS { integer | UNLIMITED }

| PCTINCREASE integer

| FREELISTS integer

| FREELIST GROUPS integer

| OPTIMAL [ integer [ K | M | G | T | P | E ] | NULL ]

| BUFFER\_POOL { KEEP | RECYCLE | DEFAULT }

}

[ INITIAL integer [ K | M | G | T | P | E ]

| NEXT integer [ K | M | G | T | P | E ]

| MINEXTENTS integer

| MAXEXTENTS { integer | UNLIMITED }

| PCTINCREASE integer

| FREELISTS integer

| FREELIST GROUPS integer

| OPTIMAL [ integer [ K | M | G | T | P | E ] | NULL ]

| BUFFER\_POOL { KEEP | RECYCLE | DEFAULT }

]

)

### Example - PERMANENT TABLESPACE

The following is a CREATE TABLESPACE statement that creates a simple permanent tablespace:

CREATE TABLESPACE tbs\_perm\_01

DATAFILE 'tbs\_perm\_01.dat'

SIZE 20M

ONLINE;

This CREATE TABLESPACE statement creates a permanent tablespace called tbs\_perm\_01 that has one data file called tbs\_perm\_01.dat.

The following is a CREATE TABLESPACE statement that creates a permanent tablespace that will extend when more space is required:

CREATE TABLESPACE tbs\_perm\_02

DATAFILE 'tbs\_perm\_02.dat'

SIZE 10M

REUSE

AUTOEXTEND ON NEXT 10M MAXSIZE 200M;

This CREATE TABLESPACE statement creates a permanent tablespace called tbs\_perm\_02 that has one data file called tbs\_perm\_02.dat. When more space is required, 10M extents will automatically be added until 200MB is reached.

The following is a CREATE TABLESPACE statement that creates a BIGFILE permanent tablespace that will extend when more space is required:

CREATE BIGFILE TABLESPACE tbs\_perm\_03

DATAFILE 'tbs\_perm\_03.dat'

SIZE 10M

AUTOEXTEND ON;

This CREATE TABLESPACE statement creates a BIGFILE permanent tablespace called tbs\_perm\_03 that has one data file called tbs\_perm\_03.dat.

## #2 - TEMPORARY TABLESPACE

A temporary tablespace contains schema objects that are stored in temp files that exist during a session.

### Syntax

The syntax for the CREATE TABLESPACE statement when creating a temporary tablespace is:

CREATE

[ SMALLFILE | BIGFILE ]

TEMPORARY TABLESPACE tablespace\_name

[ TEMPFILE { [ 'filename' | 'ASM\_filename' ]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

[ AUTOEXTEND

{ OFF

| ON [ NEXT integer [ K | M | G | T | P | E ] ]

[ MAXSIZE { UNLIMITED | integer [ K | M | G | T | P | E ] } ]

}

]

| [ 'filename | ASM\_filename'

| ('filename | ASM\_filename'

[, 'filename | ASM\_filename' ] )

]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

}

[ TABLESPACE GROUP { tablespace\_group\_name | '' } ]

[ EXTENT MANAGEMENT

{ LOCAL

[ AUTOALLOCATE | UNIFORM [ SIZE integer [ K | M | G | T | P | E ] ] ]

| DICTIONARY

} ]

**SMALLFILE**

A tablespace that contains 1,022 data or temp files (each file can be up to 4 million blocks in size). This is the most common tablespace size to create.

**BIGFILE**

A tablespace that contains only one data or temp file (this file can be up to 4 million blocks in size).

**TIP:** If you omit the SMALLFILE or BIGFILE option, the Oracle database will use the default tablespace type.

**tablespace\_name**

The name of the tablespace to create.

### Example - TEMPORARY TABLESPACE

The following is a CREATE TABLESPACE statement that creates a temporary tablespace:

CREATE TEMPORARY TABLESPACE tbs\_temp\_01

TEMPFILE 'tbs\_temp\_01.dbf'

SIZE 5M

AUTOEXTEND ON;

This CREATE TABLESPACE statement creates a temporary tablespace called tbs\_temp\_01 that has one temp file called tbs\_temp\_01.dbf.

## #3 - UNDO TABLESPACE

An undo tablespace is created to manage undo data if the Oracle database is being run in automatic undo management mode.

### Syntax

The syntax for the CREATE TABLESPACE statement when creating an undo tablespace is:

CREATE

[ SMALLFILE | BIGFILE ]

UNDO TABLESPACE tablespace\_name

[ DATAFILE { [ 'filename' | 'ASM\_filename' ]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

[ AUTOEXTEND

{ OFF

| ON [ NEXT integer [ K | M | G | T | P | E ] ]

[ MAXSIZE { UNLIMITED | integer [ K | M | G | T | P | E ] } ]

}

]

| [ 'filename | ASM\_filename'

| ('filename | ASM\_filename'

[, 'filename | ASM\_filename' ] )

]

[ SIZE integer [ K | M | G | T | P | E ] ]

[ REUSE ]

}

[ EXTENT MANAGEMENT

{ LOCAL

[ AUTOALLOCATE | UNIFORM [ SIZE integer [ K | M | G | T | P | E ] ] ]

| DICTIONARY

} ]

[ RETENTION { GUARANTEE | NOGUARANTEE } ]

**SMALLFILE**

A tablespace that contains 1,022 data or temp files (each file can be up to 4 million blocks in size). This is the most common tablespace size to create.

**BIGFILE**

A tablespace that contains only one data or temp file (this file can be up to 4 million blocks in size).

**TIP:** If you omit the SMALLFILE or BIGFILE option, the Oracle database will use the default tablespace type.

**tablespace\_name**

The name of the tablespace to create.

### Example - UNDO TABLESPACE

The following is a CREATE TABLESPACE statement that creates an undo tablespace:

CREATE UNDO TABLESPACE tbs\_undo\_01

DATAFILE 'tbs\_undo\_01.f'

SIZE 5M

AUTOEXTEND ON

RETENTION GUARANTEE;

This CREATE TABLESPACE statement creates an undo tablespace called tbs\_undo\_01 that is 5MB in size and has one data file called tbs\_undo\_01.f.