UNLIMITED

- When specified with a resource parameter, UNLIMITED indicates that a user assigned this profile can use an unlimited amount of this resource.
- When specified with a password parameter, UNLIMITED indicates that no limit has been set for the parameter.

DEFAULT

- A user assigned this profile is subject to the limit for this resource specified in the DEFAULT profile.
- The DEFAULT profile initially defines unlimited resources. You can change those limits with the ALTER PROFILE statement.

Resource Parameters:

Resource management helps in limiting the database abuse a user can cause.

For example, if a user connects to database and never runs a query then this ideal connection will take system resources like CPU.
To restrict such kind of issues, we have resource management parameters.

- **SESSIONS_PER_USER** Specify the number of concurrent sessions to which you want to limit the user.
- **CPU_PER_SESSION** Specify the CPU time limit for a session, expressed in hundredth of seconds.
- **CPU_PER_CALL** Specify the CPU time limit for a call (a parse, execute, or fetch), expressed in hundredths of seconds.
- **CONNECT_TIME** Specify the total elapsed time limit for a session, expressed in minutes.
- **IDLE_TIME** Specify the permitted periods of continuous inactive time during a session, expressed in minutes. Long-running queries and other operations are not subject to this limit.
- **LOGICAL_READS_PER_SESSION** Specify the permitted number of data blocks read in a session, including blocks read from memory and disk.
- **LOGICAL_READS_PER_CALL** Specify the permitted number of data blocks read for a call to process a SQL statement (a parse, execute, or fetch).
- **PRIVATE_SGA** Specify the amount of private space a session can allocate in the shared pool of the system global area (SGA).
- **COMPOSITE_LIMIT** Specify the total resource cost for a session, expressed in **service units**. Oracle Database calculates the total service units as a weighted sum CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION, and PRIVATE_SGA.

Password_Prameters:

The password management allows a DBA to have more control over user passwords.

FAILED_LOGIN_ATTEMPTS Specify the number of failed attempts to log in to the user account before the account is locked.

PASSWORD_LIFE_TIME Specify the number of days the same password can be used for authentication.

PASSWORD_REUSE_TIME specifies the number of days before which a password cannot be reused.

PASSWORD_REUSE_MAX specifies the number of password changes required before the current password can be reused.

PASSWORD_LOCK_TIME Specify the number of days an account will be locked after the specified number of consecutive failed login attempts.

PASSWORD_GRACE_TIME Specify the number of days after the grace period begins during which a warning is issued and login is allowed. If the password is not changed during the grace period, the password expires.

PASSWORD_VERIFY_FUNCTION The PASSWORD_VERIFY_FUNCTION clause lets a PL/SQL password complexity verification script be passed as an argument to the CREATE PROFILE statement. Oracle Database provides a default script, but you can create your own routine or use third-party software instead.

- For *function*, specify the name of the password complexity verification routine.
- Specify NULL to indicate that no password verification is performed.

Oracle Database enforces resource limits in the following ways:

- If a user exceeds the CONNECT_TIME or IDLE_TIME session resource limit, then the database rolls back the current transaction and ends the session. When the user process next issues a call, the database returns an error.

- If a user attempts to perform an operation that exceeds the limit for other session resources, then the database aborts the operation, rolls back the current statement, and immediately returns an error. The user can then commit or roll back the current transaction, and must then end the session.

- If a user attempts to perform an operation that exceeds the limit for a single call, then the database aborts the operation, rolls back the current statement, and returns an error, leaving the current transaction intact.