

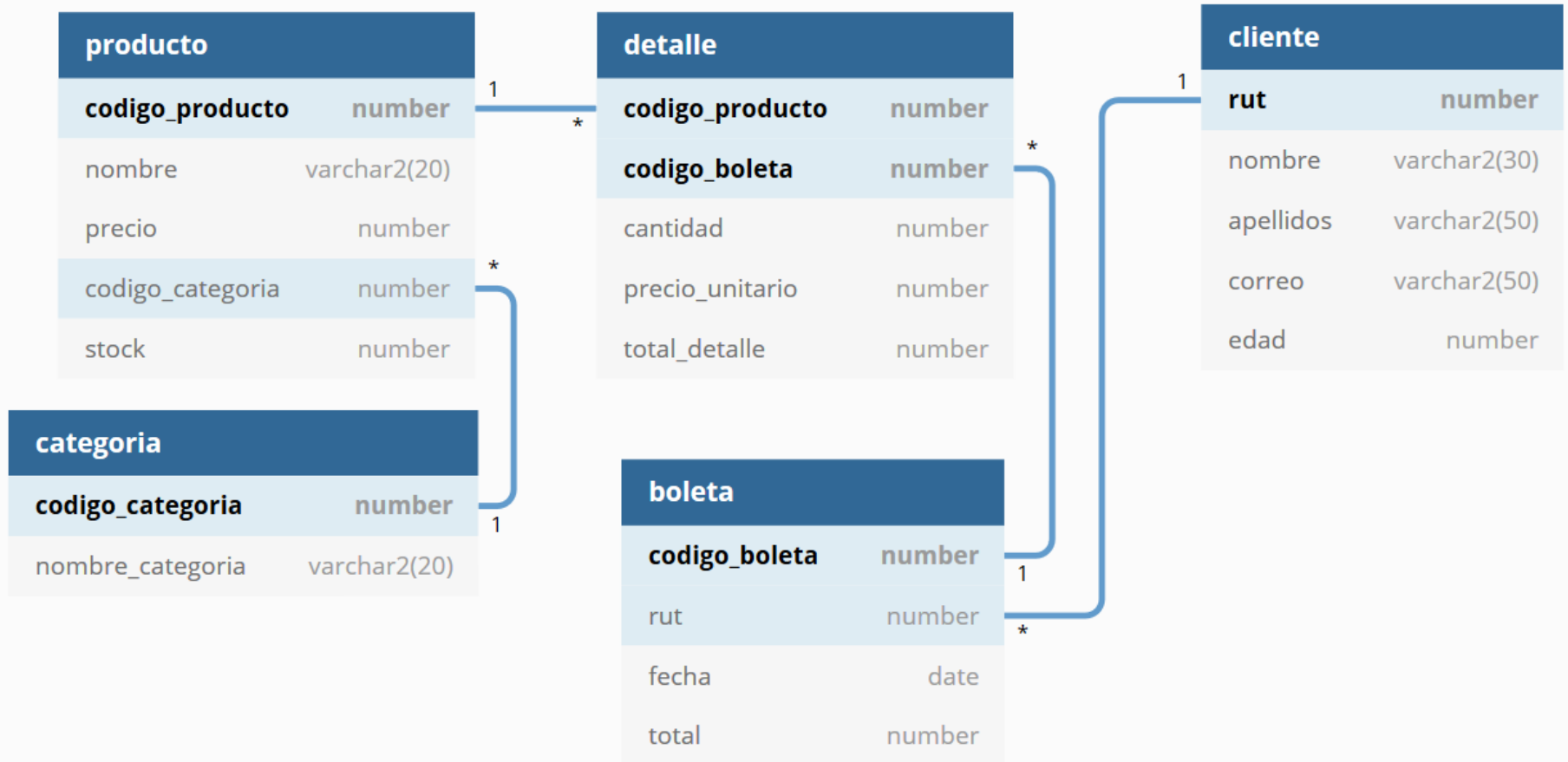
# Vistas, bloques anónimos y gestión de usuarios.

Felipe Tapia Gómez

# Programa de hoy

- Vistas
- Bloques anónimos
- Gestión de usuarios

# Modelo de trabajo de hoy



# Vistas

Las vistas son en esencia, tablas virtuales que no existen físicamente en la base de datos.

# Vistas

Las vistas son en esencia, tablas virtuales que no existen físicamente en la base de datos.

Estas se crean en base a una consulta, la cual puede ser sobre una o múltiples tablas.

# Vistas

Las vistas son en esencia, tablas virtuales que no existen físicamente en la base de datos.

Estas se crean en base a una consulta, la cual puede ser sobre una o múltiples tablas.

Una vista sirve para simplificar una consulta más compleja restringiendo al usuario lo que en realidad puede ver.

# Vistas (Sintaxis)

```
CREATE VIEW NOMBRE_VISTA AS  
SELECT COLUMNAS  
FROM TABLA  
WHERE CONDICION OPERATOR VALOR;
```

# Vistas (Sintaxis)

```
CREATE VIEW NOMBRE_VISTA AS  
SELECT COLUMNAS  
FROM TABLA  
WHERE CONDICION OPERATOR VALOR;
```

La vista básicamente es un select. Puede crear una vista de cualquier select que hayan realizado en el pasado.



# Vistas (Sintaxis)

```
CREATE VIEW NOMBRE_VISTA AS  
SELECT COLUMNAS  
FROM TABLA  
WHERE CONDICION OPERATOR VALOR;
```

La vista básicamente es un select. Puede crear una vista de cualquier select que hayan realizado en el pasado.

Más adelante en la clase veremos utilidades de las vistas al ver los permisos de usuario.

# Vistas (Ejemplo)

Como mencionamos anteriormente, la idea de las vistas es simplificar una consulta o establecer una restricción del acceso a los datos.

# Vistas (Ejemplo)

Como mencionamos anteriormente, la idea de las vistas es simplificar una consulta o establecer una restricción del acceso a los datos.

Consideremos que un usuario del sistema solicitó la consulta 19 de la clase de “Consultas de agrupación y JOIN”.

# Vistas (Ejemplo)

Muestre los productos que fueron comprados entre el 12 de Agosto y el 8 de Noviembre donde la cantidad de productos comprados fue mayor o igual a 2 productos.

La consulta debe mostrar el rut y nombre del cliente, fecha de la boleta, código, nombre, precio, cantidad, categoría y el total del producto.

# Vistas (Ejemplo)

```
SELECT C.RUT, C.NOMBRE, B.FECHA, P.CODIGO_PRODUCTO,  
P.NOMBRE, P.PRECIO, D.CANTIDAD, C.NOMBRE_CATEGORIA, D.TOTAL  
FROM CLIENTE C JOIN BOLETA B  
ON C.RUT = B.RUT JOIN DETALLE D  
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA JOIN PRODUCTO P  
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO JOIN CATEGORIA C  
ON C.CODIGO_CATEGORIA = P.CODIGO_CATEGORIA  
WHERE B.FECHA BETWEEN '12/08/2019' AND '08/11/2019'  
AND D.CANTIDAD >= 2;
```

# Vistas (Ejemplo)

```
SELECT C.RUT, C.NOMBRE, B.FECHA, P.CODIGO_PRODUCTO,  
P.NOMBRE, P.PRECIO, D.CANTIDAD, C.NOMBRE_CATEGORIA, D.TOTAL  
FROM CLIENTE C JOIN BOLETA B  
ON C.RUT = B.RUT JOIN DETALLE D  
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA JOIN PRODUCTO P  
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO JOIN CATEGORIA C  
ON C.CODIGO_CATEGORIA = P.CODIGO_CATEGORIA  
WHERE B.FECHA BETWEEN '12/08/2019' AND '08/11/2019'  
AND D.CANTIDAD >= 2;
```

Esta consulta manipula información de todas las tablas de nuestro modelo.

# Vistas (Ejemplo)

```
SELECT C.RUT, C.NOMBRE, B.FECHA, P.CODIGO_PRODUCTO,  
P.NOMBRE, P.PRECIO, D.CANTIDAD, C.NOMBRE_CATEGORIA, D.TOTAL  
FROM CLIENTE C JOIN BOLETA B  
ON C.RUT = B.RUT JOIN DETALLE D  
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA JOIN PRODUCTO P  
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO JOIN CATEGORIA C  
ON C.CODIGO_CATEGORIA = P.CODIGO_CATEGORIA  
WHERE B.FECHA BETWEEN '12/08/2019' AND '08/11/2019'  
AND D.CANTIDAD >= 2;
```

Esta consulta manipula información de todas las tablas de nuestro modelo.

Estamos entregando demasiada información hacia el usuario que utilizará la consulta.

# Vistas (Ejemplo)

```
SELECT C.RUT, C.NOMBRE, B.FECHA, P.CODIGO_PRODUCTO,  
P.NOMBRE, P.PRECIO, D.CANTIDAD, C.NOMBRE_CATEGORIA, D.TOTAL  
FROM CLIENTE C JOIN BOLETA B  
ON C.RUT = B.RUT JOIN DETALLE D  
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA JOIN PRODUCTO P  
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO JOIN CATEGORIA C  
ON C.CODIGO_CATEGORIA = P.CODIGO_CATEGORIA  
WHERE B.FECHA BETWEEN '12/08/2019' AND '08/11/2019'  
AND D.CANTIDAD >= 2;
```

Con el objetivo de garantizar seguridad y simplicidad para la consulta anterior. Utilizaremos una vista que llamaremos “productos\_filtro”.



# Vista (Ejemplo)

```
CREATE VIEW PRODUCTOS_FILTRO AS
SELECT C.RUT AS CLIENTE, C.NOMBRE AS NOMBRE_CLIENTE,
B.FECHA AS FECHA_BOLETA, P.CODIGO_PRODUCTO AS PRODUCTO,
P.NOMBRE AS NOMBRE_PRODUCTO, P.PRECIO AS PRECIO_PRODUCTO,
D.CANTIDAD AS CANTIDAD_ADQUIRIDO, D.TOTAL AS TOTAL_DETALLE,
C.NOMBRE_CATEGORIA AS CATEGORIA
FROM CLIENTE C JOIN BOLETA B
ON C.RUT = B.RUT JOIN DETALLE D
ON B.CODIGO_BOLETA = D.CODIGO_BOLETA JOIN PRODUCTO P
ON D.CODIGO_PRODUCTO = P.CODIGO_PRODUCTO JOIN CATEGORIA C
ON C.CODIGO_CATEGORIA = P.CODIGO_CATEGORIA
WHERE B.FECHA BETWEEN '12/08/2019' AND '08/11/2019'
AND D.CANTIDAD >= 2;
```

# Vista (Ejemplo)

Con la creación de la vista anterior, ahora solo es necesario entregarle al usuario la vista de la consulta que solicitó.

# Vista (Ejemplo)

Con la creación de la vista anterior, ahora solo es necesario entregarle al usuario la vista de la consulta que solicitó.

```
SELECT * FROM PRODUCTOS_FILTRO;
```

# Vista (Ejemplo)

Con la creación de la vista anterior, ahora solo es necesario entregarle al usuario la vista de la consulta que solicitó.

```
SELECT * FROM PRODUCTOS_FILTRO;
```

Esta consulta es mucho más sencilla y nos permite mantener la información a salvo.

# Vista (Ejemplo)

Con la creación de la vista anterior, ahora solo es necesario entregarle al usuario la vista de la consulta que solicitó.

```
SELECT * FROM PRODUCTOS_FILTRO;
```

Esta consulta es mucho más sencilla y nos permite mantener la información a salvo.

Más adelante veremos como vincular esto con la gestión de usuarios.

# Programación por bloques

Hasta el momento, solo hemos utilizado el lenguaje de manipulación y definición de datos.

# Programación por bloques

Hasta el momento, solo hemos utilizado el lenguaje de manipulación y definición de datos. Ahora explotaremos otro aspecto del lenguaje que es conocido como PL/SQL.

# Programación por bloques

Hasta el momento, solo hemos utilizado el lenguaje de manipulación y definición de datos. Ahora explotaremos otro aspecto del lenguaje que es conocido como PL/SQL.

Este aspecto sirve para que los programadores puedan construir bloques de código, los cuales pueden utilizarse como procedimientos o funciones.



# Programación por bloques

EL PL/SQL incluye nuevas características:

- Manejo de variables.
- Estructuras modulares.
- Estructuras de control de flujo.
- Control de excepciones.

# Programación por bloques

Dentro de la programación por bloques tenemos los siguientes tipos:

## Anonymous

```
[DECLARE]

BEGIN
  --statements

[EXCEPTION]

END;
```

## Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

## Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```

# Programación por bloques

Para comenzar con la programación por bloques, hoy trabajaremos con bloques anónimos.

# Programación por bloques

Para comenzar con la programación por bloques, hoy trabajaremos con bloques anónimos.

Los bloques anónimos son segmentos de código que pueden ejecutarse para que realicen un cálculo o resuelvan un problema.

# Programación por bloques

Para comenzar con la programación por bloques, hoy trabajaremos con bloques anónimos.

Los bloques anónimos son segmentos de código que pueden ejecutarse para que realicen un cálculo o resuelvan un problema.

Estos bloques no se guardan en la base de datos.

# Bloques anónimos

Los bloques anónimos poseen la siguiente sintaxis:

```
DECLARE  
    DECLARACION_DE_VARIABLES  
BEGIN  
    [OPERACIONES] [CONSULTAS] [ETC]  
END;
```

# Bloques anónimos

Los bloques anónimos poseen la siguiente sintaxis:

```
DECLARE  
    DECLARACION_DE_VARIABLES  
BEGIN  
    [OPERACIONES] [CONSULTAS] [ETC]  
END;
```

Para mostrar como se trabaja con un bloque anónimo, haremos un ejemplo.

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```



# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Pero, ¿Cómo podemos imprimir el “hola” en Oracle?

# Bloque anónimo (ejemplo)

```
DECLARE
  J NUMBER;
BEGIN
  FOR J IN 0..5 LOOP
    --IMPRIMIR HOLA
  END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Pero, ¿Cómo podemos imprimir el “hola” en Oracle?

Para esto, existe un comando equivalente al printf del lenguaje C o al print del lenguaje Python.

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Pero, ¿Cómo podemos imprimir el “hola” en Oracle?

Para esto, existe un comando equivalente al printf del lenguaje C o al print del lenguaje Python.

El comando es: DBMS\_OUTPUT.PUT\_LINE

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Para activar el comando DBMS\_OUTPUT.PUT\_LINE es necesario ejecutar la siguiente instrucción.

# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Para activar el comando DBMS\_OUTPUT.PUT\_LINE es necesario ejecutar la siguiente instrucción.

SET SERVEROUTPUT ON;



# Bloque anónimo (ejemplo)

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        --IMPRIMIR HOLA
    END LOOP;
END;
```

En los bloques anónimos podemos utilizar estructuras tales como IF, FOR, WHILE para trabajar la lógica de un problema.

Para este problema, queremos utilizar un FOR para imprimir la palabra HOLA.

Para activar el comando DBMS\_OUTPUT.PUT\_LINE es necesario ejecutar la siguiente instrucción.

SET SERVEROUTPUT ON;



Al ejecutar la instrucción le diremos a la base de datos, que active la impresión por consola.



# Bloque anónimo (ejemplo)

Modificando el ejemplo del FOR con la instrucción `DBMS_OUTPUT.PUT_LINE`, obtenemos el siguiente bloque anónimo.

# Bloque anónimo (ejemplo)

Modificando el ejemplo del FOR con la instrucción DBMS\_OUTPUT.PUT\_LINE, obtenemos el siguiente bloque anónimo.

```
DECLARE
    J NUMBER;
BEGIN
    FOR J IN 0..5 LOOP
        DBMS_OUTPUT.PUT_LINE('HOLA');
    END LOOP;
END;
```

El resultado que se obtiene por consola corresponde a la palabra “HOLA” repetida 5 veces.

# Bloque anónimo (ejemplo)

Podemos realizar el mismo ejemplo, utilizando la estructura WHILE

```
DECLARE
    J NUMBER;
    DETENER BOOLEAN;
BEGIN
    DETENER := FALSE;
    J := 0;
    WHILE DETENER <> TRUE LOOP
        DBMS_OUTPUT.PUT_LINE ('HOLA');
        IF J = 5 THEN
            DETENER := TRUE;
        ELSE
            J := J + 1;
        END IF;
    END LOOP;
END;
```

# Bloque anónimo (ejemplo 2)

Utilizaremos un bloque anónimo para consultar si un usuario existe en la base de datos.

```
DECLARE
    CONTADOR NUMBER;
    RUT_CONSULTAR NUMBER := 185756203;
BEGIN
    SELECT COUNT(RUT) INTO CONTADOR
    FROM CLIENTE WHERE RUT = RUT_CONSULTAR;
    DBMS_OUTPUT.PUT_LINE('EL CONTADOR ES IGUAL A: ' || CONTADOR);
    IF CONTADOR > 0 THEN
        DBMS_OUTPUT.PUT_LINE('EL USUARIO CON RUT ' || RUT_CONSULTAR || ' EXISTE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('EL USUARIO CON RUT ' || RUT_CONSULTAR || ' NO EXISTE');
    END IF;
END;
```

# Gestión de usuarios

La gestión de usuarios es importante en cualquier sistema.

Ya sea un sistema front-end o back-end, la seguridad y el control de acceso a la información siempre debe estar controlada.

# Gestión de usuarios

La gestión de usuarios es importante en cualquier sistema.

Ya sea un sistema front-end o back-end, la seguridad y el control de acceso a la información siempre debe estar controlada.

Aquí se muestra una alternativa al manejo de usuarios y control de acceso básica a los distintos objetos de la base de datos.

# CREATE USER

La instrucción CREATE USER permite crear un usuario que a su vez, actúa como base de datos.

# CREATE USER

La instrucción CREATE USER permite crear un usuario que a su vez, actúa como base de datos.

La primera vez que se instaló ORACLE, se creó el usuario “laboratorio”, el cual todos ustedes utilizan.



# CREATE USER

La instrucción CREATE USER permite crear un usuario que a su vez, actúa como base de datos.

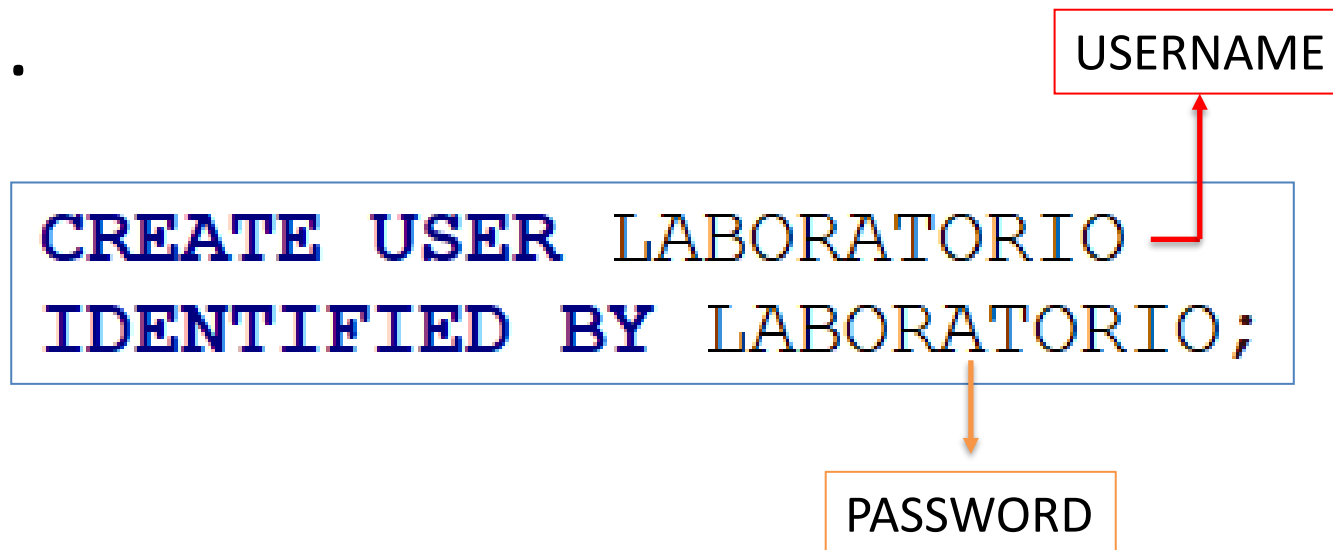
La primera vez que se instaló ORACLE, se creó el usuario “laboratorio”, el cual todos ustedes utilizan.

```
CREATE USER LABORATORIO  
IDENTIFIED BY LABORATORIO;
```

# CREATE USER

La instrucción CREATE USER permite crear un usuario que a su vez, actúa como base de datos.

La primera vez que se instaló ORACLE, se creó el usuario “laboratorio”, el cual todos ustedes utilizan.



# CREATE ROLE

Posterior a su creación, se creó un ROL que poseía los permisos necesarios para que “laboratorio” pueda funcionar.

```
CREATE ROLE LABDCI;
```

# Asignación de permisos

Una vez creado el ROL, es necesario darle permisos.

# Asignación de permisos

Una vez creado el ROL, es necesario darle permisos.

Para asignar y quitar un permiso al ROL, se utilizan las siguientes instrucciones.

```
GRANT PERMISO TO ROL/USUARIO;
```

```
REVOKE PERMISO FROM ROL/USUARIO;
```

# Permisos básicos utilizados

Privilegio	Tipo objeto	Descripción
DELETE	Tabla	Permite al usuario eliminar desde una tabla.
EXECUTE	Procedimiento o función	Permite al usuario ejecutar un procedimiento o función.
INSERT	Tabla o Vista	Permita al usuario insertar sobre una tabla.
SELECT	Tabla, secuencia, vista	Permite al usuario consultar sobre una tabla, secuencia o vista.
UPDATE	Tabla	Permite al usuario actualizar una tabla.

[Más info de privilegios, aquí](#)

# Permisos básicos utilizados

Privilegio	Tipo objeto	Descripción
DELETE	Tabla	Permite al usuario eliminar desde una tabla.
EXECUTE	Procedimiento o función	Permite al usuario ejecutar un procedimiento o función.
INSERT	Tabla o Vista	Permita al usuario insertar sobre una tabla.
SELECT	Tabla, secuencia, vista	Permite al usuario consultar sobre una tabla, secuencia o vista.
UPDATE	Tabla	Permite al usuario actualizar una tabla.

[Más info de privilegios, aquí](#)

Permiso básico para realizar conexión a la base de datos:

```
GRANT CREATE SESSION TO LABDCI;
```

# Ejercicio simple de usuarios

Hemos visto los pasos básicos para crear un usuario y asignarle permisos por medio de un rol.



# Ejercicio simple de usuarios

Hemos visto los pasos básicos para crear un usuario y asignarle permisos por medio de un rol.

Antes de comenzar con el ejercicio, debemos comprender el funcionamiento interno del sistema de usuarios de Oracle.

# Espacios de trabajo

Oracle posee espacios de trabajo donde los usuarios pueden trabajar.

# Espacios de trabajo

Oracle posee espacios de trabajo donde los usuarios pueden trabajar.

Cada vez que creamos un usuario, es necesario vincularlo a un espacio de trabajo.

# Espacios de trabajo

Oracle posee espacios de trabajo donde los usuarios pueden trabajar.

Cada vez que creamos un usuario, es necesario vincularlo a un espacio de trabajo.

Si bien es posible crear nuevos espacios de trabajo, utilizaremos los que Oracle ya trae por defecto.

# Espacios de trabajo

Oracle trae por defecto 5 espacios de trabajo.

# Espacios de trabajo

Oracle trae por defecto 5 espacios de trabajo.

```
TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
```

# Espacios de trabajo

Oracle trae por defecto 5 espacios de trabajo.

```
TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
```

Cada vez que creamos un usuario, es necesario asociarlo a un espacio de trabajo.

# Espacios de trabajo

Oracle trae por defecto 5 espacios de trabajo.

```
TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
```

Cada vez que creamos un usuario, es necesario asociarlo a un espacio de trabajo.

Para el ejemplo, utilizaremos el espacio de trabajo “SYSTEM” para todos los usuarios.



# Simulación de un sistema

Vamos a suponer un sistema donde existe un administrador y un cliente.

# Simulación de un sistema

Vamos a suponer un sistema donde existe un administrador y un cliente.

La idea es crear un usuario “administrador”, el cual posea todos los privilegios.

# Simulación de un sistema

Vamos a suponer un sistema donde existe un administrador y un cliente.

La idea es crear un usuario “administrador”, el cual posea todos los privilegios.

Y además, crear un usuario “cliente”, que tenga ciertos privilegios sobre “administrador”.

# Simulación de un sistema

Vamos a suponer un sistema donde existe un administrador y un cliente.

La idea es crear un usuario “administrador”, el cual posea todos los privilegios.

Y además, crear un usuario “cliente”, que tenga **ciertos** privilegios sobre “administrador”.

Para crear estos usuarios, primero, nos conectaremos al motor de base de datos como super usuario.

# Conexión como super usuario.

Iniciar la consola de Oracle

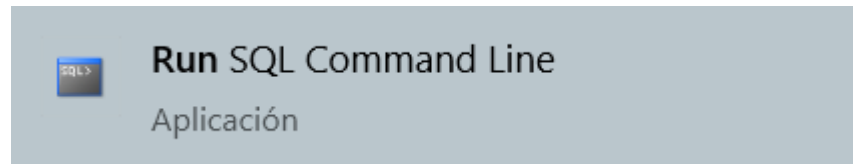


**Run SQL Command Line**

Aplicación

# Conexión como super usuario.

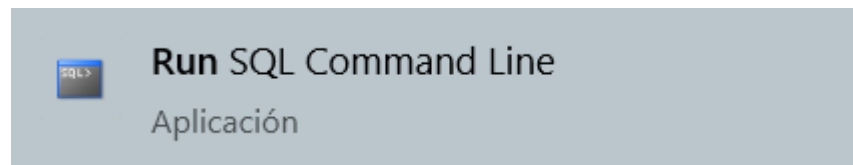
Iniciar la consola de Oracle



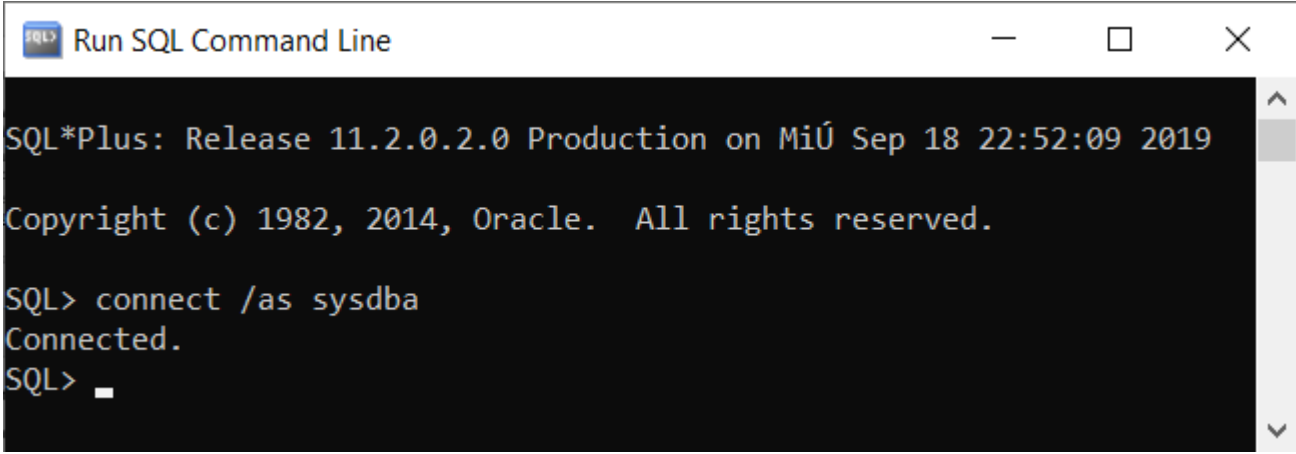
Una vez que aparezca la consola, conectarse como super usuario.

# Conexión como super usuario.

## Iniciar la consola de Oracle



Una vez que aparezca la consola, conectarse como super usuario.



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on MiÚ Sep 18 22:52:09 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> _
```

# Creación primer usuario

Ya con nuestra consola en funcionamiento y conectados como super usuario, realizaremos las siguientes operaciones.



# Creación primer usuario

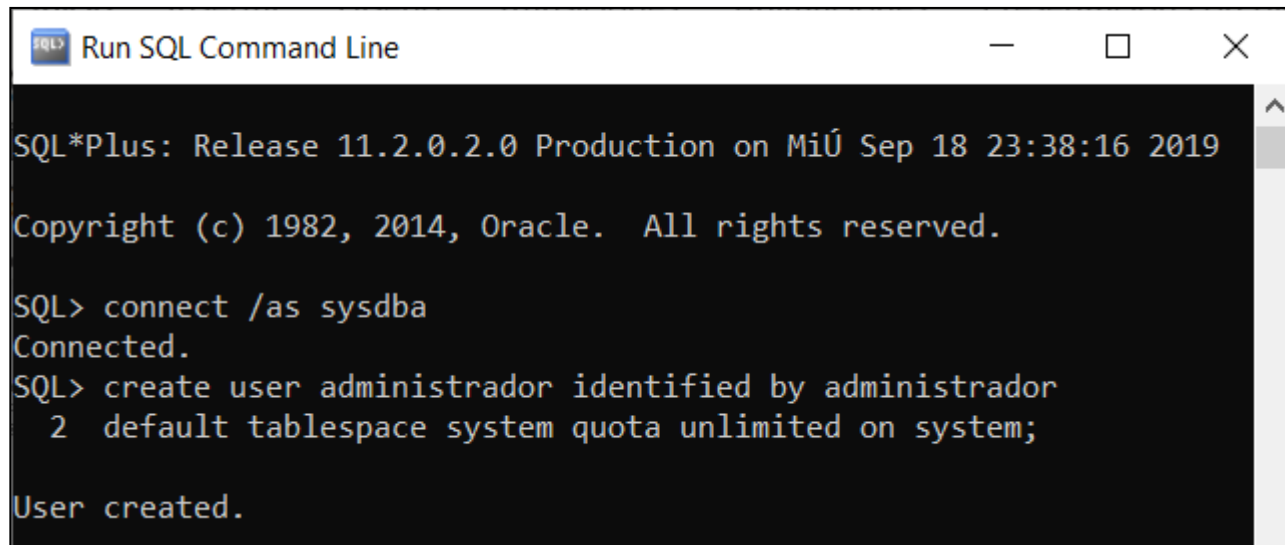
Ya con nuestra consola en funcionamiento y conectados como super usuario, realizaremos las siguientes operaciones.

- Crear el usuario administrador.
- Crear el rol administrador.
- Dar privilegios al rol administrador.
- Cargar el rol administrador al usuario administrador.

# Creación primer usuario

Crear al usuario administrador.

```
CREATE USER ADMINISTRADOR  
IDENTIFIED BY ADMINISTRADOR  
DEFAULT TABLESPACE SYSTEM  
QUOTA UNLIMITED ON SYSTEM;
```

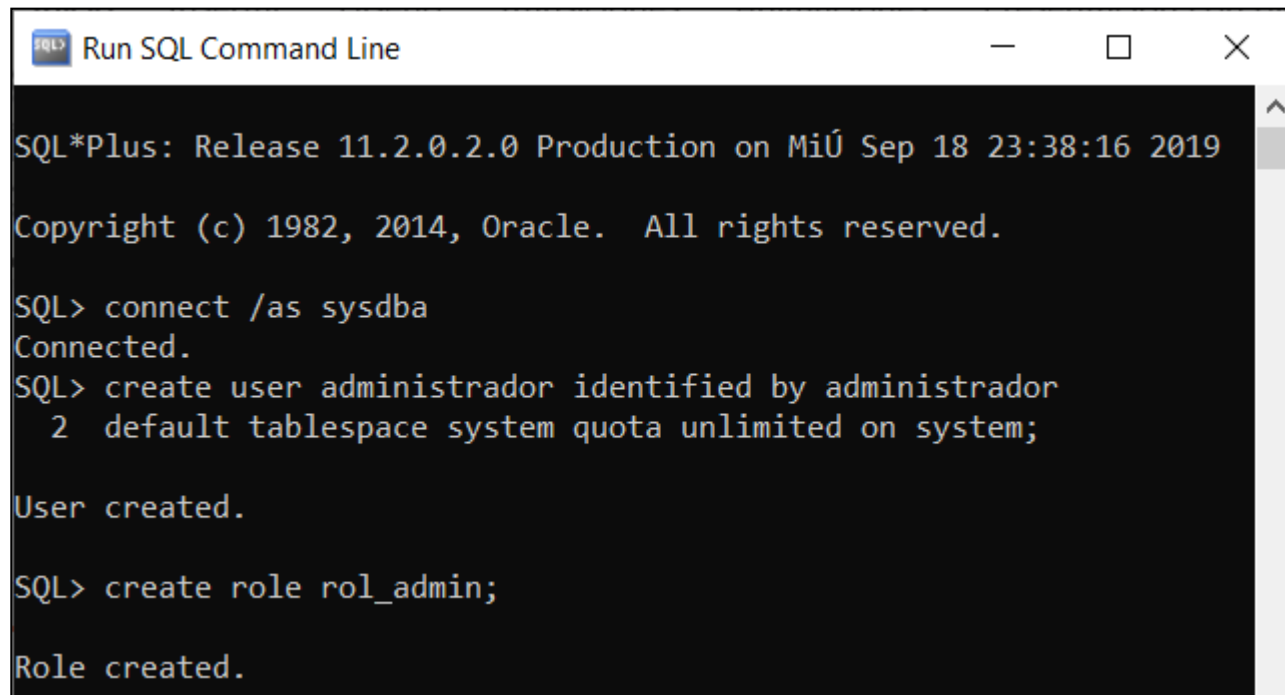


```
SQL> Run SQL Command Line  
SQL*Plus: Release 11.2.0.2.0 Production on MiÚ Sep 18 23:38:16 2019  
Copyright (c) 1982, 2014, Oracle. All rights reserved.  
SQL> connect /as sysdba  
Connected.  
SQL> create user administrador identified by administrador  
2 default tablespace system quota unlimited on system;  
User created.
```

# Creación primer usuario

Crear el rol administrador.

```
CREATE ROLE ROL_ADMIN;
```



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on MiÚ Sep 18 23:38:16 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user administrador identified by administrador
      2  default tablespace system quota unlimited on system;

User created.

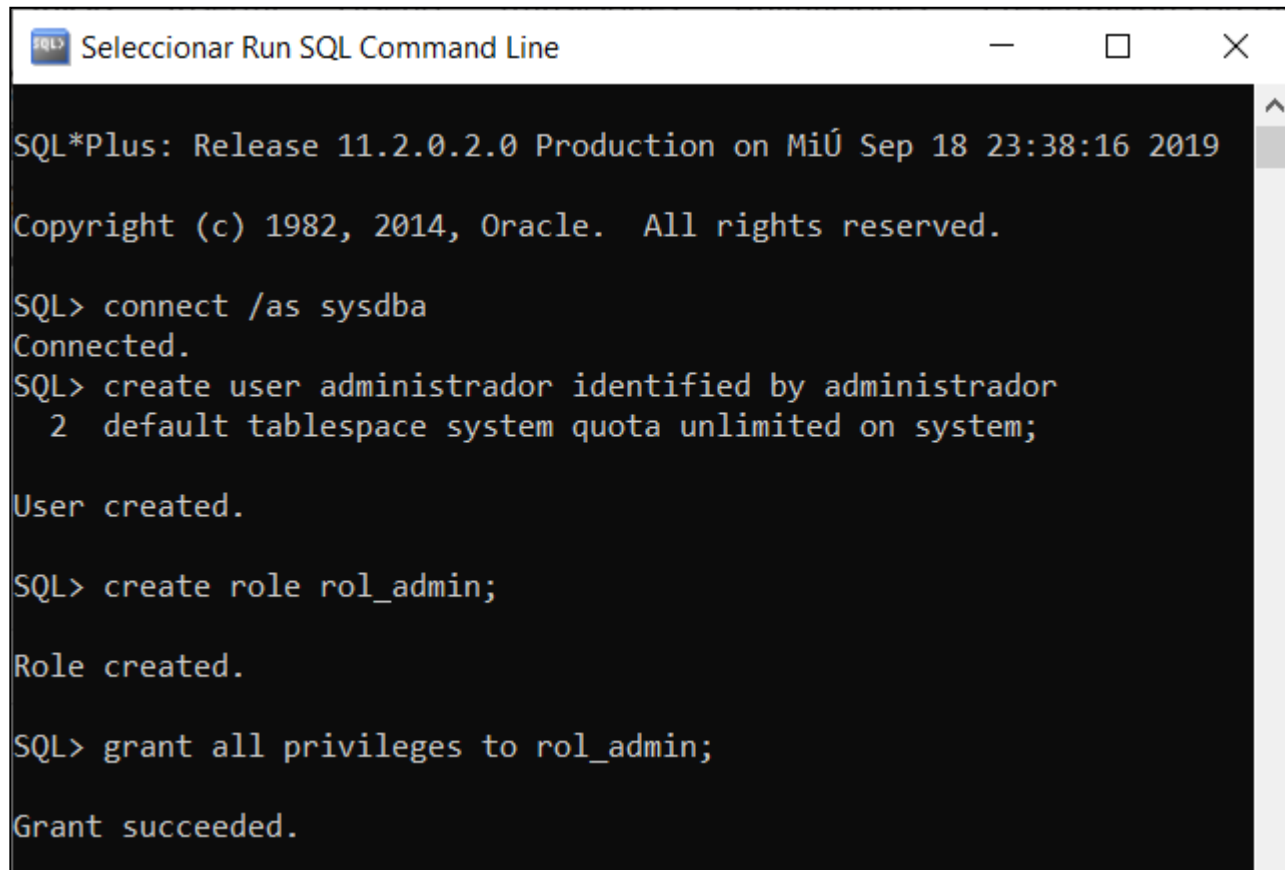
SQL> create role rol_admin;

Role created.
```

# Creación primer usuario

Dar privilegios al rol administrador.

```
GRANT ALL PRIVILEGES TO ROL_ADMIN;
```



```
SQL> Seleccionar Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on MiÚ Sep 18 23:38:16 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user administrador identified by administrador
      2  default tablespace system quota unlimited on system;

User created.

SQL> create role rol_admin;

Role created.

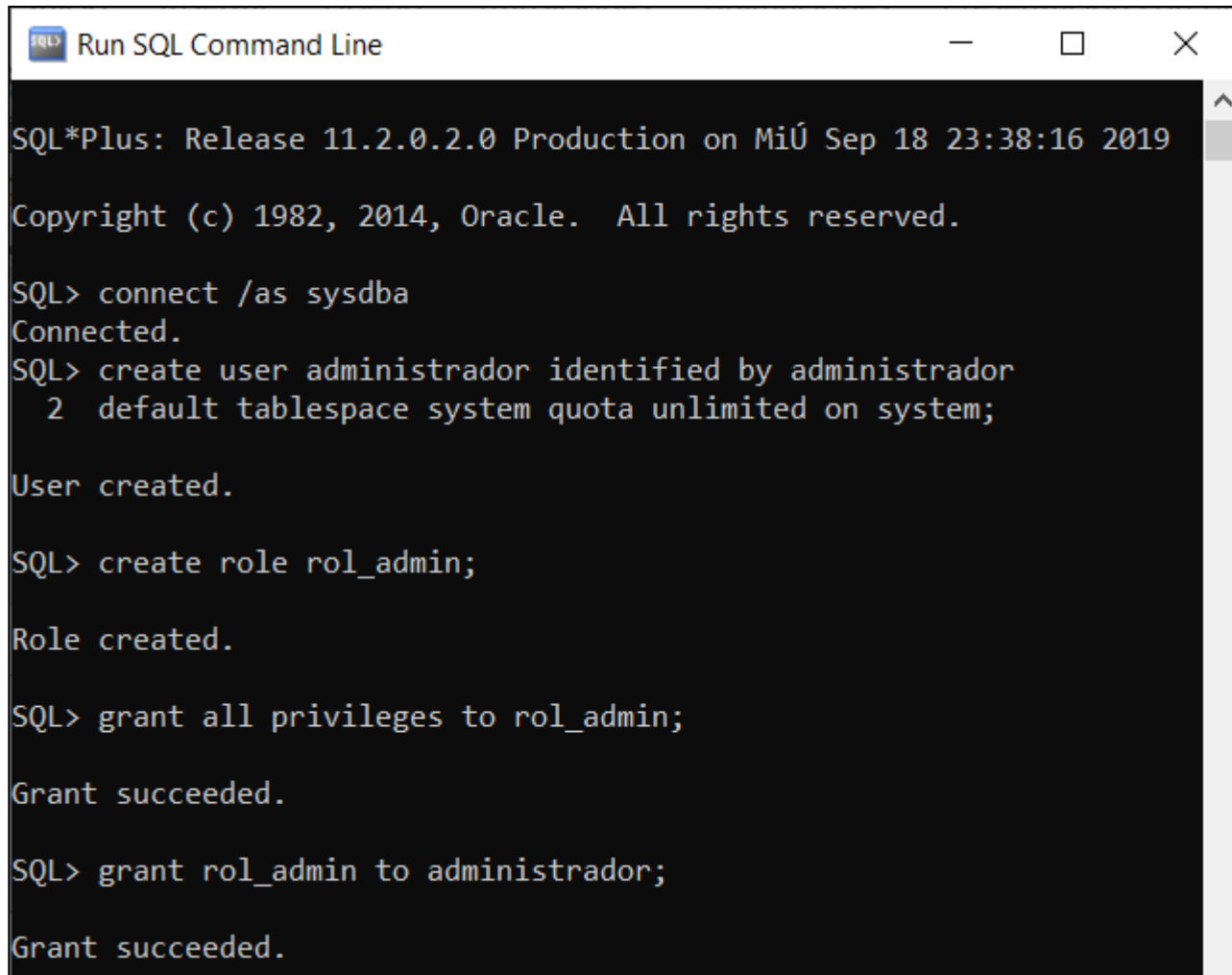
SQL> grant all privileges to rol_admin;

Grant succeeded.
```

# Creación primer usuario

Cargar el rol administrador al usuario administrador.

```
GRANT ROL_ADMIN TO ADMINISTRADOR;
```



```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on MiÚ Sep 18 23:38:16 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user administrador identified by administrador
      2  default tablespace system quota unlimited on system;

User created.

SQL> create role rol_admin;

Role created.

SQL> grant all privileges to rol_admin;

Grant succeeded.

SQL> grant rol_admin to administrador;

Grant succeeded.
```

# Creación primer usuario

En resumen, todos los comandos ingresados.

```
CREATE USER ADMINISTRADOR  
IDENTIFIED BY ADMINISTRADOR  
DEFAULT TABLESPACE SYSTEM  
QUOTA UNLIMITED ON SYSTEM;  
  
CREATE ROLE ROL_ADMIN;  
  
GRANT ALL PRIVILEGES TO ROL_ADMIN;  
  
GRANT ROL_ADMIN TO ADMINISTRADOR;
```

# Creación primer usuario

En resumen, todos los comandos ingresados.

```
CREATE USER ADMINISTRADOR  
IDENTIFIED BY ADMINISTRADOR  
DEFAULT TABLESPACE SYSTEM  
QUOTA UNLIMITED ON SYSTEM;  
  
CREATE ROLE ROL_ADMIN;  
  
GRANT ALL PRIVILEGES TO ROL_ADMIN;  
  
GRANT ROL_ADMIN TO ADMINISTRADOR;
```

Con el ingreso de todos los comandos anteriores, el usuario “administrador” tiene privilegios sobre el espacio de trabajo “SYSTEM”.

# Verificando el primer usuario

Ya creado el usuario “administrador”, realizamos la prueba en el SQL Developer.

Nueva / Seleccionar Conexión a Base de Datos

Nombre de Cone... Detalles de Cone...

Name administrador

Tipo de Base de Datos Oracle

**Información de usuario** Usuario de Proxy

Tipo de autenticación Por defecto

Usuario administrador Rol valor por defecto

Contraseña ..... ☐ Guardar Contraseña

Tipo de Conexión Básico

**Detalles** Avanzado

Nombre del Host localhost

Puerto 1521

☒ SID xe

☐ Nombre del Servicio

Estado: Correcto

Ayuda Guardar Borrar Probar Conectar Cancelar



# Verificando el primer usuario

Si todo fue realizado correctamente, la conexión debió ser exitosa.

Ahora puede crear una tabla en nuestro nuevo usuario.

# Verificando el primer usuario

Si todo fue realizado correctamente, la conexión debió ser exitosa.

Ahora puede crear una tabla en nuestro nuevo usuario.

```
CREATE TABLE TEST (  
    NUMERO NUMBER  
);
```

Si todo fue realizado correctamente, se debió crear la tabla.

# Creación segundo usuario

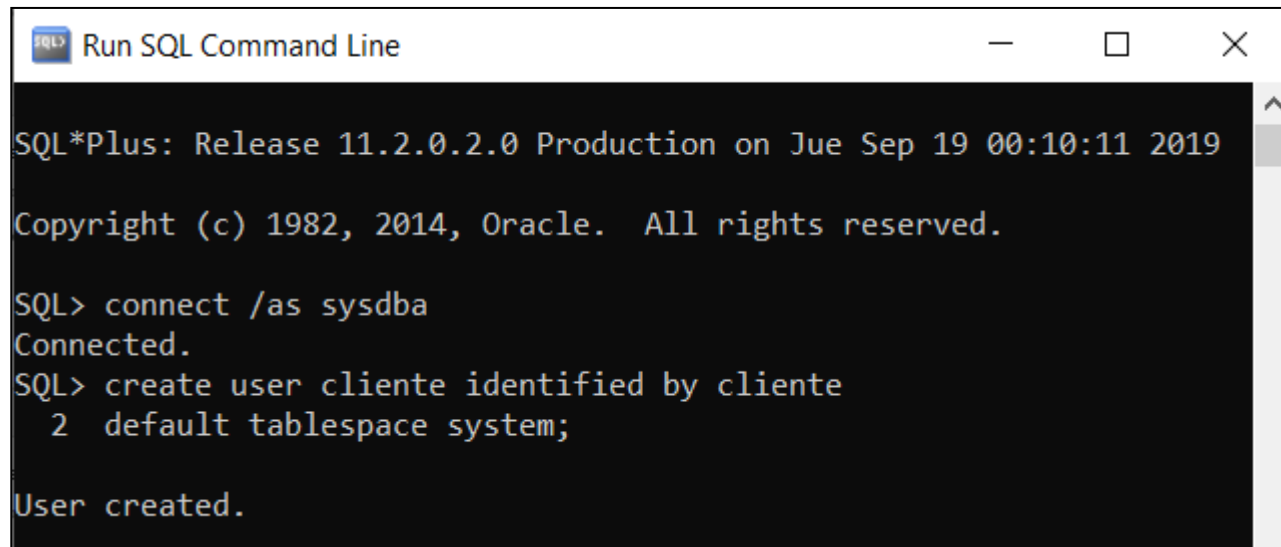
Suponiendo que no ha cerrado la consola, ahora procederemos a crear nuestro segundo usuario. Las operaciones son las siguientes:

- Crear el usuario cliente.
- Crear el rol cliente.
- Dar privilegios al rol cliente.
- Cargar el rol cliente al usuario cliente.

# Creación segundo usuario

Crear el usuario cliente.

```
CREATE USER CLIENTE  
IDENTIFIED BY CLIENTE  
DEFAULT TABLESPACE SYSTEM;
```

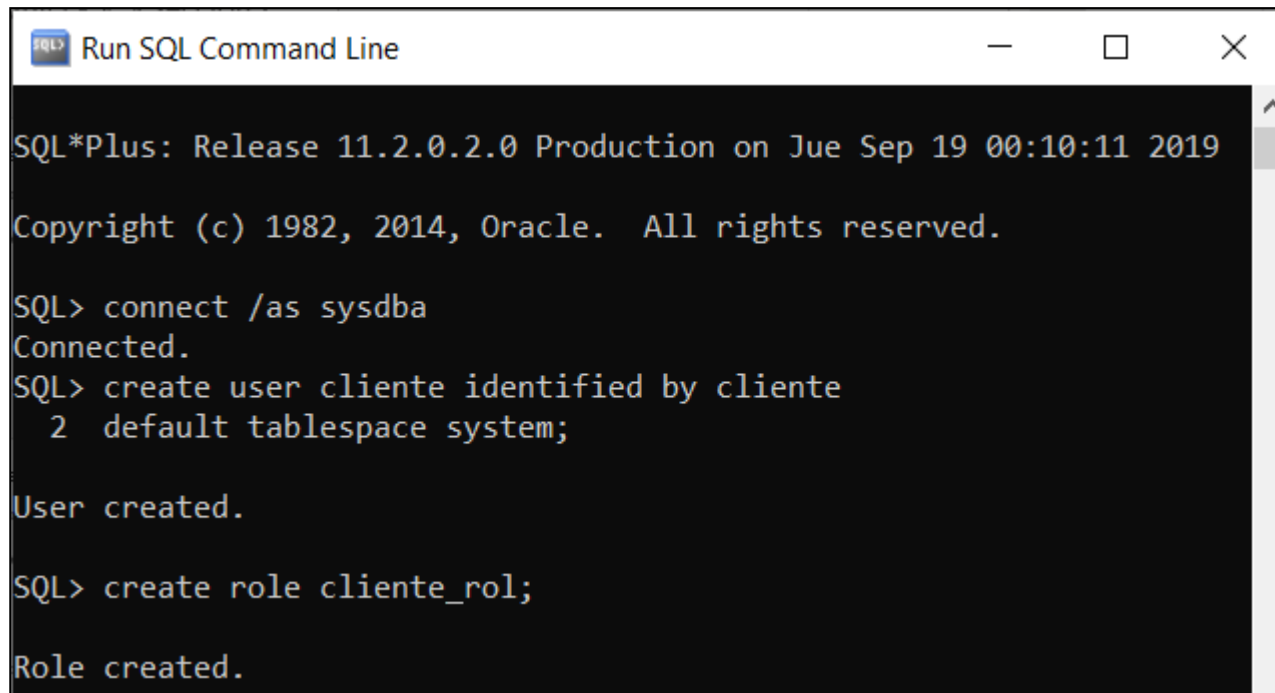


```
SQL> Run SQL Command Line  
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 00:10:11 2019  
Copyright (c) 1982, 2014, Oracle. All rights reserved.  
  
SQL> connect /as sysdba  
Connected.  
SQL> create user cliente identified by cliente  
2 default tablespace system;  
  
User created.
```

# Creación segundo usuario

Crear el rol cliente.

```
CREATE ROLE CLIENTE_ROL;
```



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 00:10:11 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user cliente identified by cliente
      2 default tablespace system;

User created.

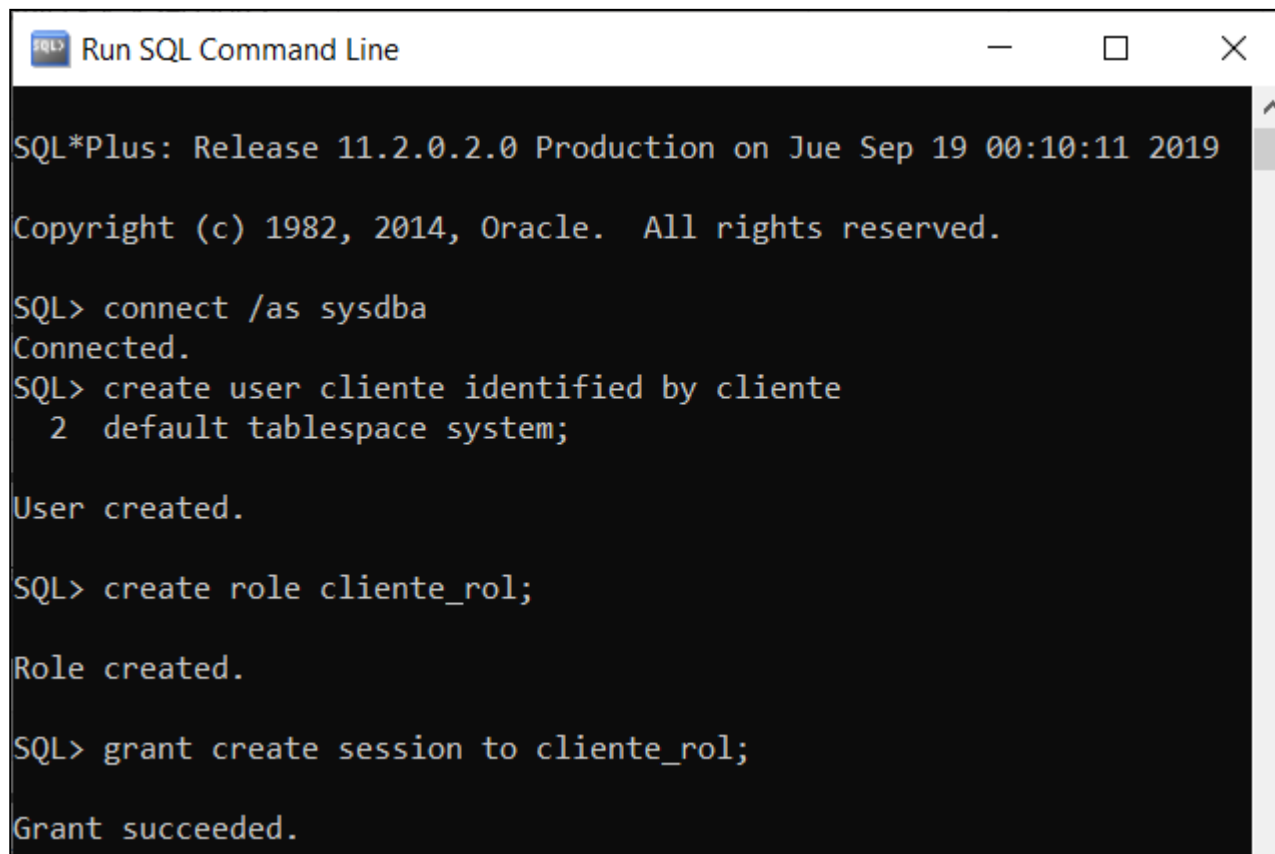
SQL> create role cliente_rol;

Role created.
```

# Creación segundo usuario

Dar **solo el privilegio de conexión** al rol cliente.

```
GRANT CREATE SESSION TO CLIENTE_ROL;
```



```
SQL> Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 00:10:11 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user cliente identified by cliente
      2  default tablespace system;

User created.

SQL> create role cliente_rol;

Role created.

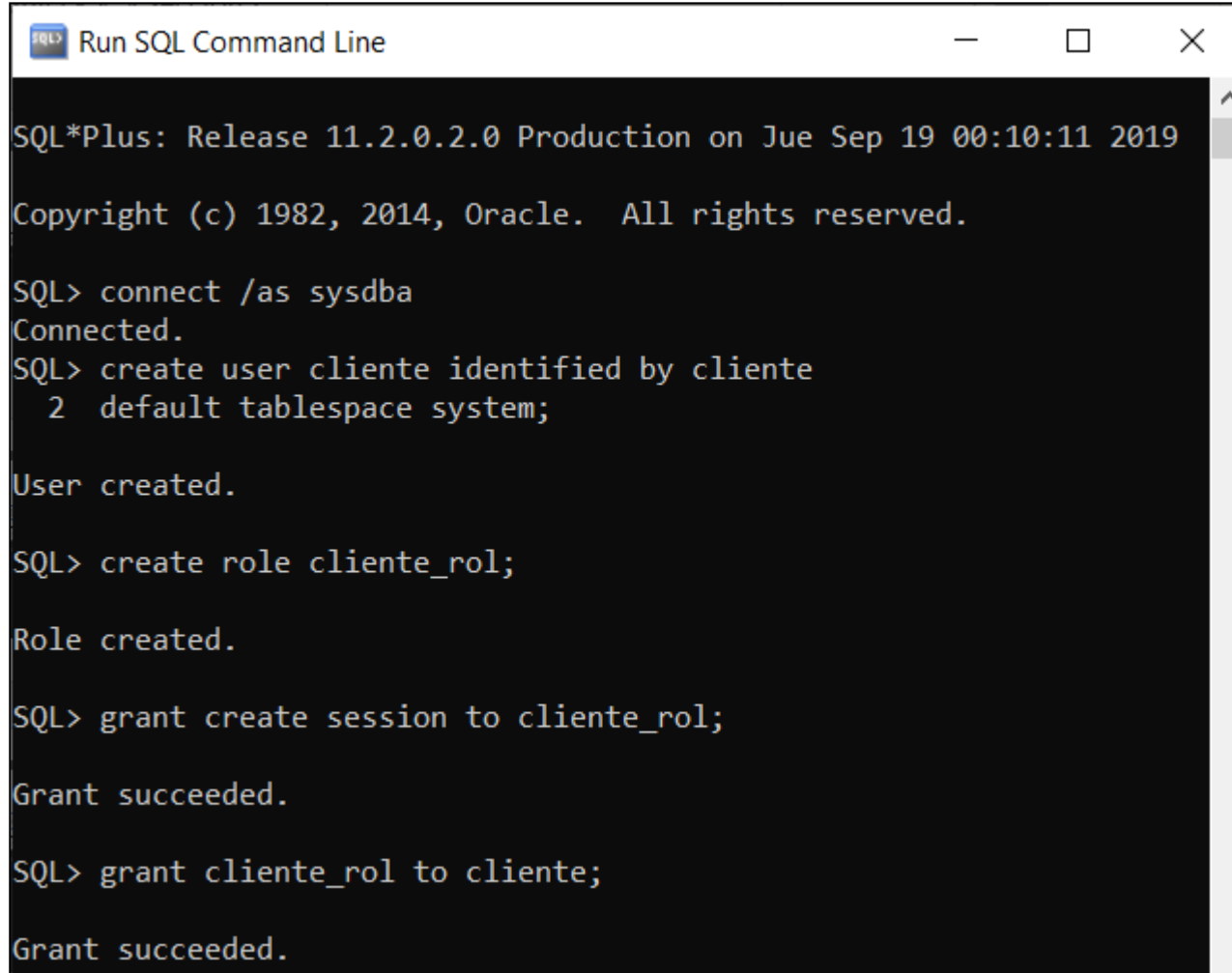
SQL> grant create session to cliente_rol;

Grant succeeded.
```

# Creación segundo usuario

Cargar el rol cliente al usuario cliente.

```
GRANT CLIENTE_ROL TO CLIENTE;
```



```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 00:10:11 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect /as sysdba
Connected.
SQL> create user cliente identified by cliente
      2  default tablespace system;

User created.

SQL> create role cliente_rol;

Role created.

SQL> grant create session to cliente_rol;

Grant succeeded.

SQL> grant cliente_rol to cliente;

Grant succeeded.
```

# Creación segundo usuario

En resumen, todos los comandos ingresados.

```
CREATE USER CLIENTE  
IDENTIFIED BY CLIENTE  
DEFAULT TABLESPACE SYSTEM;  
  
CREATE ROLE CLIENTE_ROL;  
  
GRANT CREATE SESSION TO CLIENTE_ROL;  
  
GRANT CLIENTE_ROL TO CLIENTE;
```



# Creación segundo usuario

En resumen, todos los comandos ingresados.

```
CREATE USER CLIENTE  
IDENTIFIED BY CLIENTE  
DEFAULT TABLESPACE SYSTEM;  
  
CREATE ROLE CLIENTE_ROL;  
  
GRANT CREATE SESSION TO CLIENTE_ROL;  
  
GRANT CLIENTE_ROL TO CLIENTE;
```

Con el ingreso de todos los comandos anteriores, el usuario “cliente” solo tiene el privilegio para conectarse al espacio de trabajo “SYSTEM”.

# Verificando el segundo usuario

Ya creado el usuario “cliente”, realizamos la prueba en el SQL Developer.

Nueva / Seleccionar Conexión a Base de Datos

Nombre de Cone...	Detalles de Cone...
administrador	administrador@//...

Name: cliente

Tipo de Base de Datos: Oracle

**Información de usuario** | Usuario de Proxy

Tipo de autenticación: Por defecto

Usuario: cliente

Contraseña: .....

Rol: valor por defecto

☐ Guardar Contraseña

Tipo de Conexión: Básico

**Detalles** | Avanzado

Nombre del Host: localhost

Puerto: 1521

☒ SID: xe

☐ Nombre del Servicio

Estado: Correcto

Ayuda | Guardar | Borrar | Probar | Conectar | Cancelar

# Síntesis

Hasta el momento, hemos creado dos usuarios.

# Síntesis

Hasta el momento, hemos creado dos usuarios.

- Administrador siendo el usuario con los privilegios para crear tablas, procedimientos, funciones, etc.

# Síntesis

Hasta el momento, hemos creado dos usuarios.

- Administrador siendo el usuario con los privilegios para crear tablas, procedimientos, funciones, etc.
- Cliente siendo el usuario que solo tiene permiso de conexión.

¿Qué sigue?

# ¿Qué sigue?

Ahora, debemos iniciar sesión sobre la cuenta de administrador.

# ¿Qué sigue?

Ahora, debemos iniciar sesión sobre la cuenta de administrador.

Nuestro objetivo es manipular la base de datos como administrador, pero también, otorgar ciertos privilegios al usuario cliente, para que también tenga acceso a la base de datos.



# Ejemplo permisos

Anteriormente, creamos en administrador, una tabla llamada TEST.

# Ejemplo permisos

Anteriormente, creamos en administrador, una tabla llamada TEST.

Si bien administrador y cliente están sobre el mismo espacio de trabajo, cliente solo tiene acceso a la conexión.

# Ejemplo permisos

Anteriormente, creamos en administrador, una tabla llamada TEST.

Si bien administrador y cliente están sobre el mismo espacio de trabajo, cliente solo tiene acceso a la conexión.

Por lo tanto, el cliente no puede ver la tabla TEST que está presente en el espacio de trabajo SYSTEM.

# Ejemplo permisos

Ahora es nuestro trabajo indicar qué puede hacer el cliente con la tabla TEST.

# Ejemplo permisos

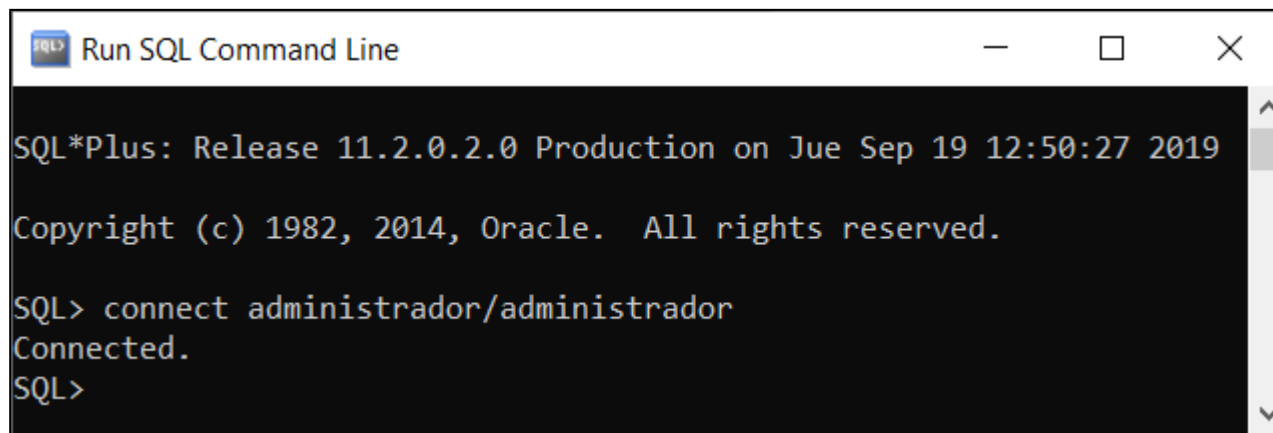
Ahora es nuestro trabajo indicar qué puede hacer el cliente con la tabla TEST.

Para esto, nos dirigimos a la consola de Oracle, para así, iniciar sesión como administrador.

# Ejemplo permisos

Ahora es nuestro trabajo indicar qué puede hacer el cliente con la tabla TEST.

Para esto, nos dirigimos a la consola de Oracle, para así, iniciar sesión como administrador.

A screenshot of a Windows-style window titled "Run SQL Command Line". The window has a black background with white text. The text inside the window shows the SQL\*Plus startup sequence: "SQL\*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:50:27 2019", "Copyright (c) 1982, 2014, Oracle. All rights reserved.", and the user prompt "SQL>". The user has entered the command "connect administrador/administrador", and the response "Connected." is shown. The prompt "SQL>" is visible again at the bottom.

```
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:50:27 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect administrador/administrador
Connected.
SQL>
```

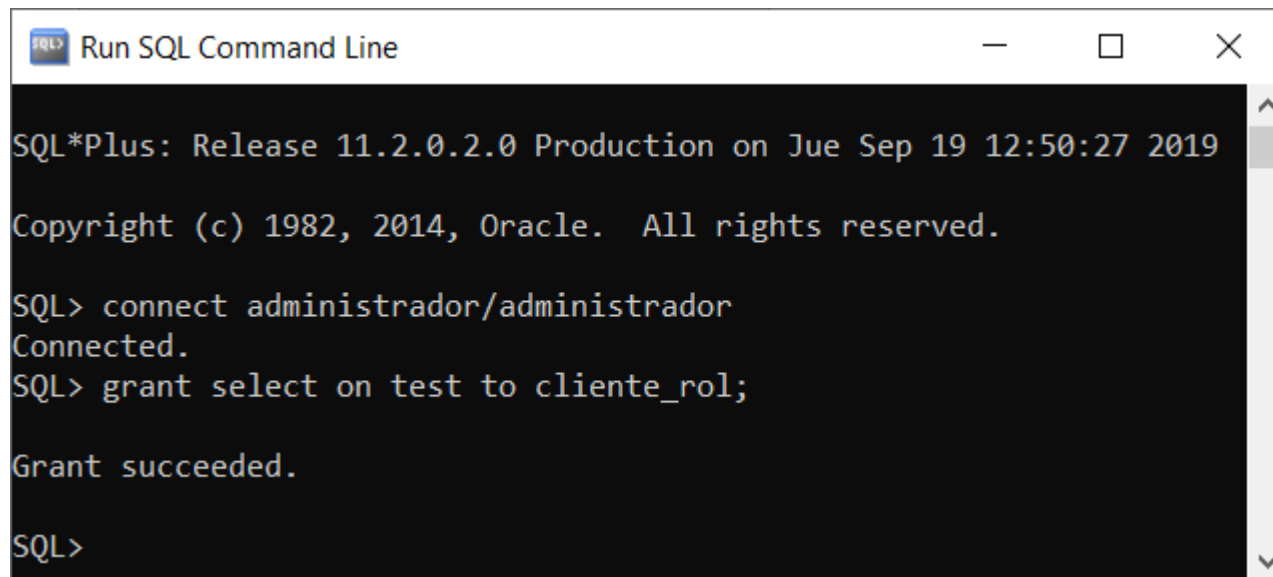
# Ejemplo permisos

Aquí, utilizaremos los permisos que están en la diapositiva 54.

# Ejemplo permisos

Aquí, utilizaremos los permisos que están en la diapositiva 54.

Otorgaremos al rol cliente, el permiso de hacer un SELECT sobre la tabla TEST.



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:50:27 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect administrador/administrador
Connected.
SQL> grant select on test to cliente_rol;

Grant succeeded.

SQL>
```

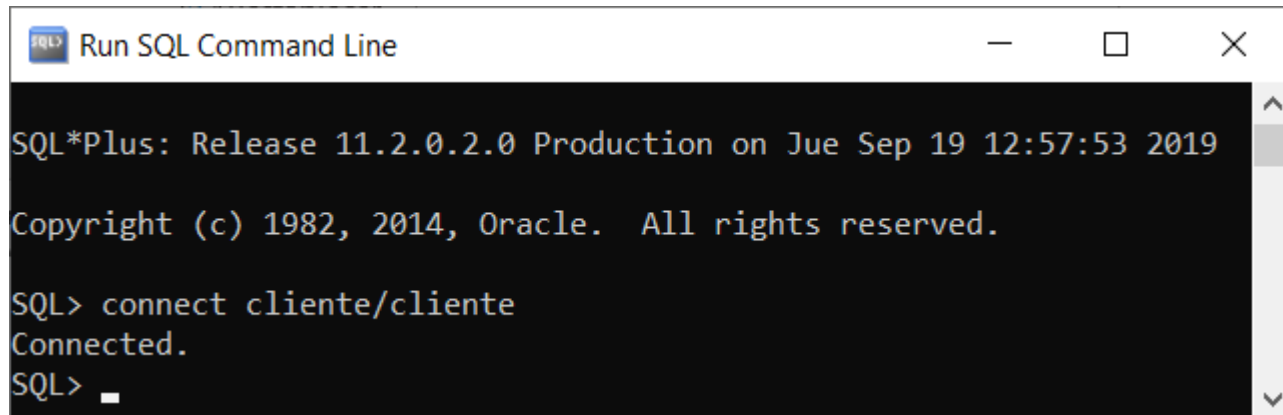


# Ejemplo permisos

Ahora, es necesario verificar que el permiso fue otorgado con éxito.

# Ejemplo permisos

Ahora, es necesario verificar que el permiso fue otorgado con éxito. Para esto, abriremos otra consola y nos conectamos como el usuario cliente.



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:57:53 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect cliente/cliente
Connected.
SQL> _
```

# Ejemplo permisos

En la consola, ejecutamos la instrucción:

# Ejemplo permisos

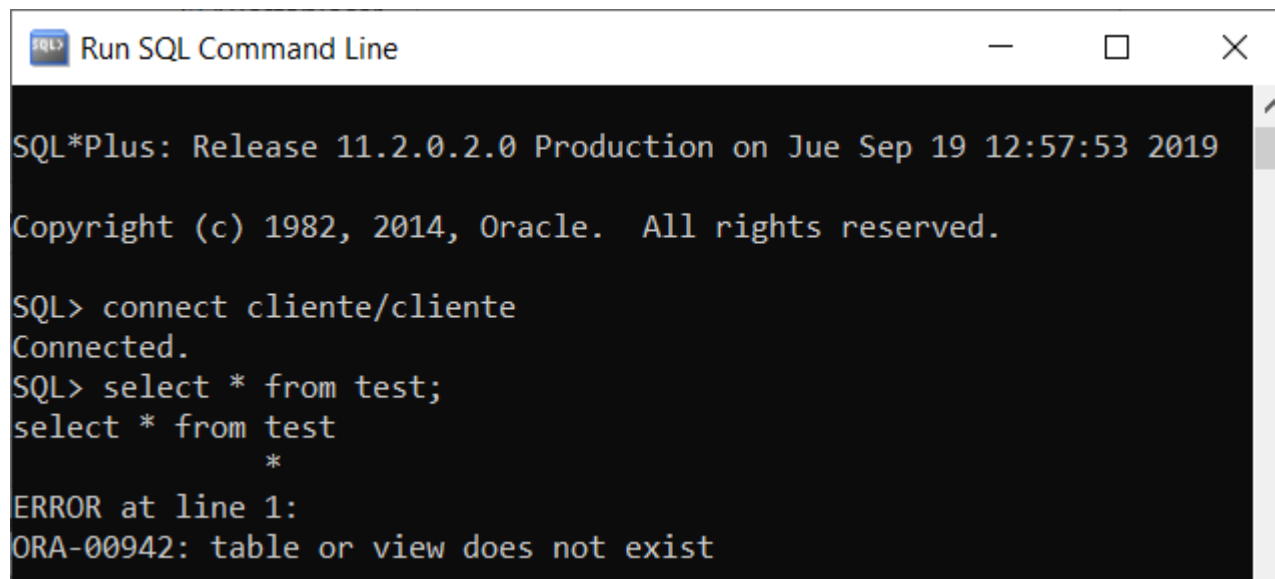
En la consola, ejecutamos la instrucción:

```
SELECT * FROM TEST;
```

# Ejemplo permisos

En la consola, ejecutamos la instrucción:

```
SELECT * FROM TEST;
```

A screenshot of a SQL command line window titled "Run SQL Command Line". The window has a black background with white text. The text inside the window shows the following sequence of events: 1. The SQL\*Plus release information: "SQL\*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:57:53 2019". 2. The copyright notice: "Copyright (c) 1982, 2014, Oracle. All rights reserved." 3. The user connects: "SQL> connect cliente/cliente" followed by "Connected." 4. The user enters a query: "SQL> select \* from test;" followed by "select \* from test" on the next line. 5. An error occurs: "ERROR at line 1:" followed by "ORA-00942: table or view does not exist".

```
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 12:57:53 2019  
Copyright (c) 1982, 2014, Oracle. All rights reserved.  
  
SQL> connect cliente/cliente  
Connected.  
SQL> select * from test;  
select * from test  
*  
ERROR at line 1:  
ORA-00942: table or view does not exist
```

El resultado indicará que la tabla no existe.

# Ejemplo permisos

El error anterior es correcto porque no hemos especificado el espacio de trabajo donde se encuentra la tabla TEST.

# Ejemplo permisos

El error anterior es correcto porque no hemos especificado el espacio de trabajo donde se encuentra la tabla TEST.

Cabe mencionar que el permiso fue otorgado desde el usuario “administrador”, por lo que, para consultar la tabla TEST, debemos hacer referencia al “usuario”. “nombre\_tabla”.

# Ejemplo permisos

El error anterior es correcto porque no hemos especificado el espacio de trabajo donde se encuentra la tabla TEST.

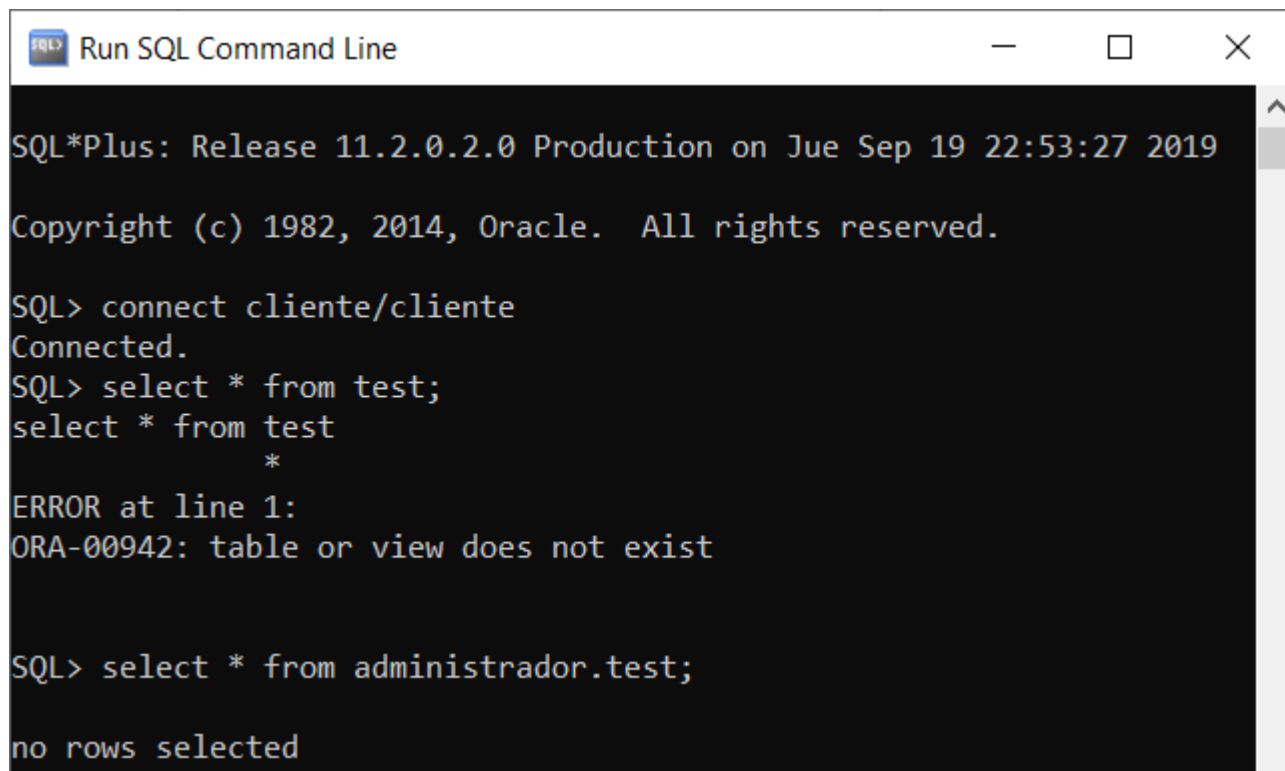
Cabe mencionar que el permiso fue otorgado desde el usuario “administrador”, por lo que, para consultar la tabla TEST, debemos hacer referencia al “usuario”. “nombre\_tabla”.

```
SELECT * FROM ADMINISTRADOR.TEST;
```



# Ejemplo permisos

Lo mencionado anteriormente se traduce en lo siguiente:



```
SQL> Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Jue Sep 19 22:53:27 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect cliente/cliente
Connected.
SQL> select * from test;
select * from test
              *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> select * from administrador.test;

no rows selected
```

# Ejemplo permisos

Como sabemos, cada tabla, procedimiento, función, vista y trigger será creado sobre el usuario “administrador”.

# Ejemplo permisos

Como sabemos, cada tabla, procedimiento, función, vista y trigger será creado sobre el usuario “administrador”.

Por lo tanto, para utilizar estos objetos desde el usuario “cliente”, siempre habrá que utilizar el formato: administrador.Objeto

# Ejemplo permisos

Como sabemos, cada tabla, procedimiento, función, vista y trigger será creado sobre el usuario “administrador”.

Por lo tanto, para utilizar estos objetos desde el usuario “cliente”, siempre habrá que utilizar el formato: administrador.Objeto

```
SELECT * FROM ADMINISTRADOR.TEST;
```

# Ejercicios

Utilizando solamente bloques anónimos realice los siguientes ejercicios.

- Dado un número, determinar si es par o impar.
- Escriba un programa que simule una calculadora básica, este puede realizar operación de suma, resta, multiplicación y división.

# Ejercicios

- Escriba un programa que entregue la edad del usuario a partir de su fecha de nacimiento.
- Escriba un programa que muestre la tabla de multiplicar del 1 al 10 de un número determinado.
- Escriba un programa que pida al usuario dos números enteros, y luego entregue la suma de todos los números que están entre ellos. Por ejemplo, si los números son 1 y 7, debe entregar como resultado  $2 + 3 + 4 + 5 + 6 = 20$ .

# Ejercicios

Desarrolle un programa que calcule el dígito verificador de un RUT.

Para calcular el dígito verificador, se deben realizar los siguiente pasos:

- Obtener el rut sin puntos, sin guión ni dígito verificador.
- Invertir el número. (e.g: de 20101234 a 43210102).
- Multiplicar los dígitos por la secuencia 2, 3, 4, 5, 6, 7, si es que se acaban los números, se debe comenzar de nuevo, por ejemplo, con 43210102:

$$4 \times 2 + 3 \times 3 + 2 \times 4 + 1 \times 5 + 0 \times 6 + 1 \times 7 + 0 \times 2 + 2 \times 3 = 43$$

- Al resultado obtenido, es decir, 43, debemos sacarle el módulo 11, es decir:

$$43 \% 11 = 10$$

- Con el resultado obtenido en el paso anterior, debemos restarlo de 11:  
$$11 - 10 = 1$$
- Finalmente, el dígito verificador será el obtenido en la resta: 20101234-1.

