

Taller 2 - Bitácora

| | |
|----------------|---|
| 🕒 Created | @November 13, 2023 6:42 PM |
| 📁 Class | Admin Bases |
| 👤 Estudiante 1 | Oscar Armando Calderón Arguera 00090822 |
| 👤 Estudiante 2 | Juan Neftaly Castellanos Hernández 00182222 |

Visite la versión web de esta guía: <https://periwinkle-telescope-913.notion.site/Taller-2-Bit-cora-584264b91a7f4db4acc8e75621f00a1f?pvs=4>

Introducción

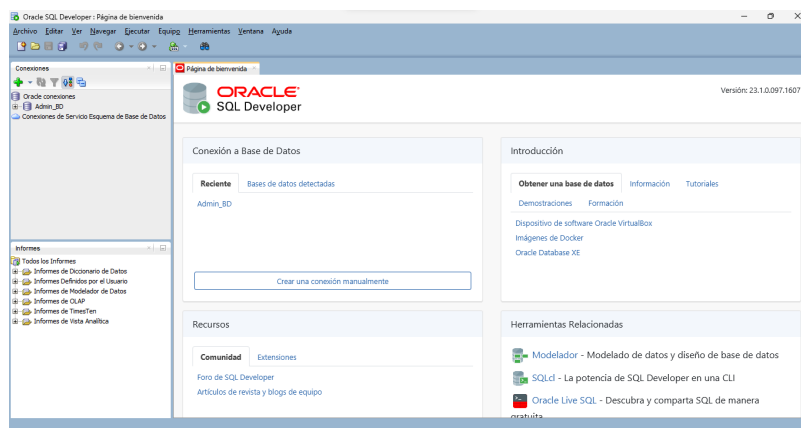
Le damos la bienvenida a la bitácora de Administración de Bases de Datos, el presente documento tiene como principal objetivo proporcionar una guía detallada para la implementación de un sistema de base de datos, y que este se pueda replicar en cualquier dispositivo que cuente con el motor de base de datos adecuado. Esperamos que esta guía no solo sea de utilidad para la resolución del paradigma que se nos presenta, sino que también pueda ser tomada como un recurso educativo.

Primeros pasos

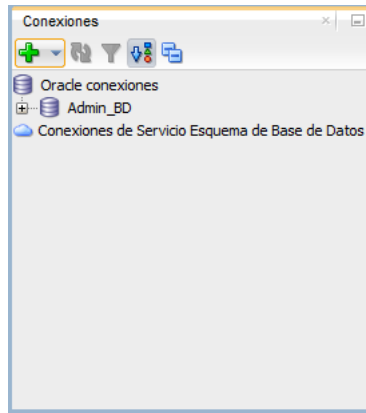
Para comenzar esta guía es necesario que el dispositivo en el que usted vaya a realizar la implementación tenga instalado Oracle Database, asumimos que para esta guía usted ya cuenta con el motor listo para su ejecución, sino es así le invitamos a que lo instale. Puede seguir la siguiente guía de instalación:

https://ecampus.uca.edu.sv/pluginfile.php/484476/mod_resource/content/0/Gu%C3%ADa%20de%20instalaci

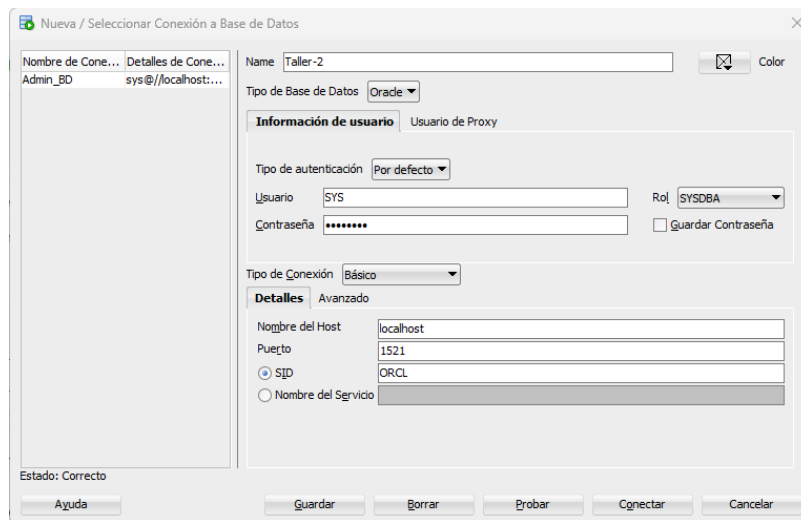
Al iniciar el motor de base de datos, usted verá algo parecido a esto:



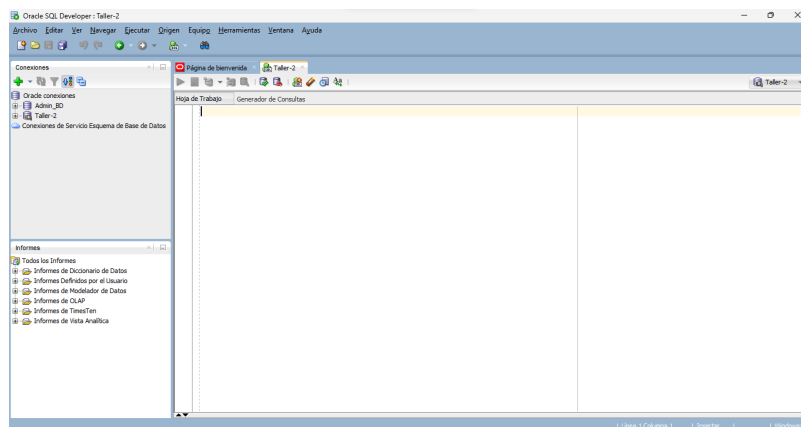
Iniciaremos una nueva conexión para preparar la base de datos, para esto es necesario que demos clic en el símbolo más que aparece en la esquina superior izquierda:



Al crear una nueva conexión nos aparecerá una ventanilla la cual debemos llenar con los siguientes datos. Si comprobamos la



Y listo, ya estaremos en el entorno de Oracle, felicidades, podemos comenzar a trabajar:



A continuación comenzaremos a trabajar en el editor de Oracle, para esta siguiente parte nos basaremos en el script adjunto **Taller2 - AdminBD**

Creación de los espacios de trabajo

El script de trabajo contiene los pasos a seguir en comentarios y los respectivos bloques de SQL a ejecutar. **Esta parte debe ser desde la conexión SYS**

La siguiente línea de código nos permite crear usuarios, roles y perfiles:

```
-- Comando necesario para la creación del usuario
ALTER SESSION SET "_ORACLE_SCRIPT" = TRUE;
```

A continuación crearemos los espacios de trabajo, los Tablespace para guardar los datos:

```
-- Creando tablespaces desde el usuario SYS
CREATE TABLESPACE t_dbadmin
  DATAFILE 'C:\demo\t_dbamin.dbf' SIZE 10M;

CREATE TABLESPACE t_empleados
  DATAFILE 'C:\demo\t_empleados.dbf' SIZE 10M;

CREATE TABLESPACE t_estudiantes
  DATAFILE 'C:\demo\t_estudiantes.dbf' SIZE 10M;
```

Podemos verificar que los tablespaces se han creados con la siguiente vista:

```
SELECT * FROM DBA_TABLESPACES;
SELECT * FROM DBA_DATA_FILES;
```

A continuación crearemos un usuario administrador y a este le asignaremos un perfil que cumpla con `ora12c_strong_verify_function` y otras restricciones que consideramos necesarias.

Crearemos el perfil en el editor en Oracle, para colocar las restricciones necesarias, siempre en la conexión SYS:

```
-- Creando perfil para el usuario
CREATE PROFILE strong_profile LIMIT
  PASSWORD_LIFE_TIME 60
  PASSWORD_REUSE_TIME 365
  PASSWORD_REUSE_MAX 5
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_VERIFY_FUNCTION ora12c_strong_verify_function;
```

A continuación creamos el usuario, con los parámetros que nos han solicitado y le asignamos el perfil `strong_profile` que creamos anteriormente:

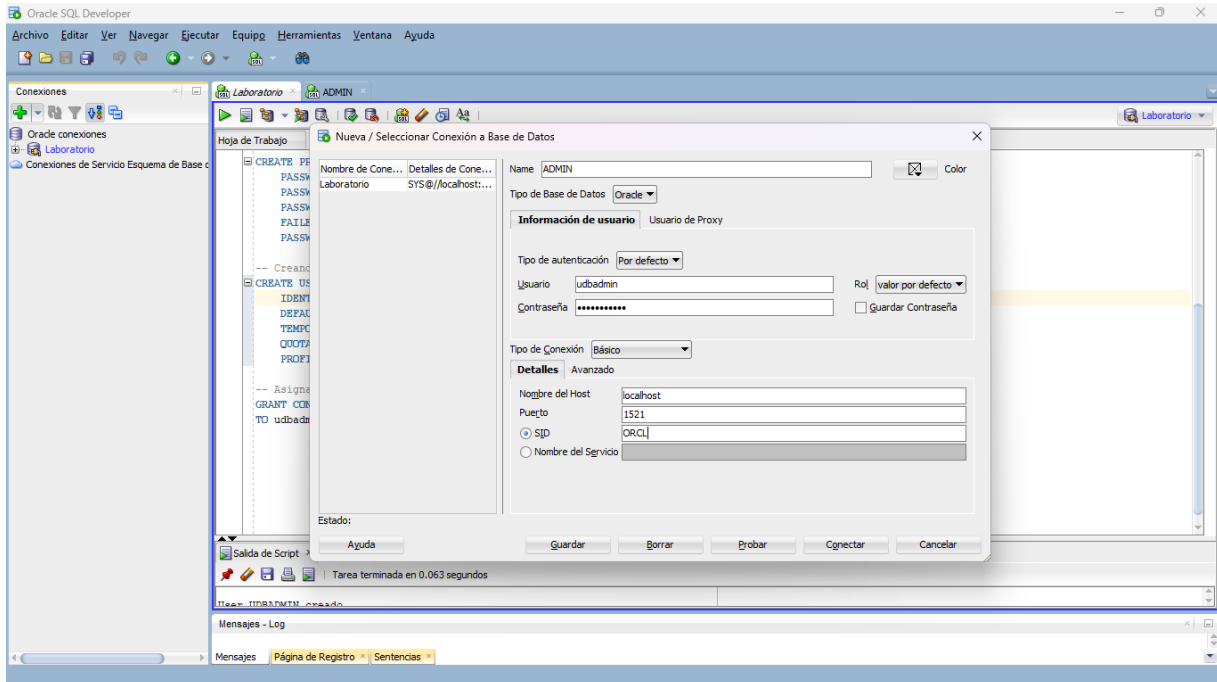
```
-- Creando el usuario administrador de la base de datos
CREATE USER udbadmin
  IDENTIFIED BY Za3$Xy7#Qw2
  DEFAULT TABLESPACE t_dbadmin
  TEMPORARY TABLESPACE temp
  QUOTA UNLIMITED ON t_dbadmin
  PROFILE strong_profile;
```

Por ultimo asignamos los privilegios a nuestro usuario:

```
-- Asignando privilegios al usuario udbadmin
GRANT CONNECT, CREATE TABLE
TO udbadmin;
```

Creación de los roles correspondientes a los roles de trabajo

Abriremos una conexión con el usuario `udbadmin`, el cual fue creado previamente.



Desde esta conexión crearemos el modelo de datos que se encuentra en el archivo `database.sql` es de

Como se espera que todos los usuarios puedan tener cierto contacto con la base de datos, mas no todos deberían tener la misma capacidad de modificar esta, es por eso que, según las buenas prácticas de la Administración de Bases de Datos, crearemos una estructura de roles, esto desde una conexión del usuario `SYS`.

Rol Administrativo

Para el rol administrativo se desea que pueda conectarse a la base de datos, y que pueda insertar, seleccionar, borrar y actualizar sobre las tablas `EMPLEADO` y `ESTUDIANTE` para esto ejecutaremos desde la conexión `SYS` las siguientes líneas de código:

```
-- Rol administrativo
CREATE ROLE administrativo;

-- Asignando privilegios al rol administrativo
GRANT CREATE SESSION TO administrativo;
GRANT INSERT, SELECT, DELETE, UPDATE ON EMPLEADO TO administrativo;
GRANT INSERT, SELECT, DELETE, UPDATE ON ESTUDIANTE TO administrativo;
```

Rol Coordinador

Para el rol coordinador se desea que pueda conectarse a la base de datos, y que pueda insertar, seleccionar, borrar y actualizar sobre las tablas `ALUMNO`, `MATERIA`, `SECCION`, Y `LISTA_ESTUDIANTES`. Se desea que solo pueda hacer `SELECT` sobre las siguientes tablas `EVALUACION` Y

NOTA. Además se desea que el coordinador solo pueda ver la lista de empleados que sean docentes.

Primero creamos el Rol

```
CREATE ROLE coordinador;
```

Después de eso, ejecutaremos unas sentencias para la solución del ultimo requisito, el cual es que el coordinador solo pueda ver a los empleados que son docentes, es decir, que tiene su id de tipo de empleado en el valor 2, esto lo podemos ver en la tabla de empleados:

```
CREATE TABLE EMPLEADOS (  
  carnet CHAR(8) PRIMARY KEY,  
  nombre VARCHAR2(256),  
  correo VARCHAR2(256),  
  id_tipo_empleado INT -- 0: administrativo, 1: coordinador, 2: docente  
);
```

crearemos una vista para acceder a estos datos, y lo haremos de la siguiente manera:

```
CREATE OR REPLACE VIEW DOCENTES (Carnet, Nombre_Empleado, email) AS  
SELECT carnet, nombre, correo FROM empleados  
WHERE id_tipo_empleado = 2  
WITH CHECK OPTION;
```

En esta vista llamada **DOCENTES** extraemos el carnet, nombre y correo del empleado que son los datos que nos interesan, poniendo la condición de que solo los EMPLEADOS con el id 2 sean mostrados

```
-- Asignando privilegios al rol coordinador  
GRANT CREATE SESSION TO coordinador;  
GRANT SELECT, UPDATE ON ALUMNO TO coordinador;  
GRANT INSERT, SELECT, DELETE, UPDATE ON MATERIA TO coordinador;  
GRANT INSERT, SELECT, DELETE, UPDATE ON SECCION TO coordinador;  
GRANT INSERT, SELECT, DELETE, UPDATE ON LISTA_ESTUDIANTES TO coordinador;  
GRANT SELECT ON EVALUACION TO coordinador;  
GRANT SELECT ON NOTA TO coordinador;  
  
-- Asignando privilegio a coordinador sobre vista  
GRANT SELECT ON DOCENTES TO coordinador;
```

Con la creación de esta vista estamos asegurando que solo se muestren los datos deseados, además de tener la capacidad de proteger datos.

Rol Docente

Para el rol docente se pide lo siguiente: capacidad de insertar, seleccionar, borrar y actualizar datos en las tablas **EVALUACION** y **NOTA**. Y que pueda seleccionar los datos sobre las tablas **ALUMNO**, **SECCION** y **LISTA_ESTUDIANTES**.

```
-- Rol docente  
CREATE ROLE docente;  
  
-- Asignando privilegios a docente  
GRANT CREATE SESSION TO DOCENTE;  
GRANT INSERT, SELECT, DELETE, UPDATE ON EVALUACION TO DOCENTE;  
GRANT INSERT, SELECT, DELETE, UPDATE ON NOTA TO DOCENTE;  
GRANT SELECT ON ALUMNO TO DOCENTE;  
GRANT SELECT ON SECCION TO DOCENTE;  
GRANT SELECT ON LISTA_ESTUDIANTES TO DOCENTE;
```

Rol Estudiante

Para el rol estudiante, queremos que este pueda conectarse a la base, y ver los datos de las tablas EVALUACION y NOTA, para eso ejecutaremos las siguientes líneas:

```
-- Rol estudiante
CREATE ROLE estudiante;

-- Asignando privilegios al rol estudiante
GRANT CREATE SESSION TO ESTUDIANTE;
GRANT SELECT ON EVALUACION TO ESTUDIANTE;
GRANT SELECT ON NOTA TO ESTUDIANTE;
```

Tratamiento de los datos

En esta sección veremos como insertaremos los datos provistos a la base de datos, tenemos que tomar en cuenta lo siguiente:

- Administrativos, coordinadores y docentes se guardan en la tabla EMPLEADO
- Estudiantes se guardan en la tabla ESTUDIANTE

NOTA: En el caso de los empleados, se han definido las siguientes etiquetas que deberán respetarse a la hora de insertar los usuarios: 0: administrativo, 1: coordinador, 2: docente. Esta etiqueta se almacenará en la columna `id_tipo_empleado` de la tabla EMPLEADO.

Procesando los datos

Para la ejecución de la tarea, se ha seleccionado el lenguaje de programación Java en conjunto con el entorno de desarrollo IntelliJ IDEA. El código correspondiente está disponible en el siguiente repositorio, facilitando así el acceso y la revisión del mismo.

GitHub - Arguera123/Taller2-ADB-Solucion-Archivos

Contribute to Arguera123/Taller2-ADB-Solucion-Archivos development by creating an account on GitHub.

<https://github.com/Arguera123/Taller2-ADB-Solucion-Archivos.git>

Arguera123/Taller2-ADB-Solucion-Archivos



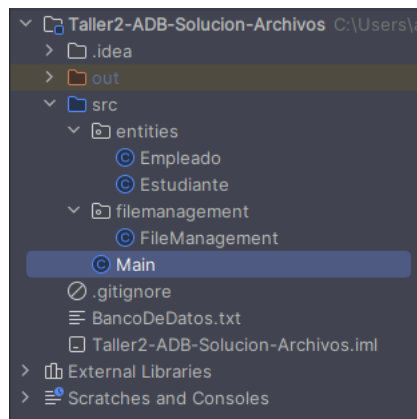
2 Contributors

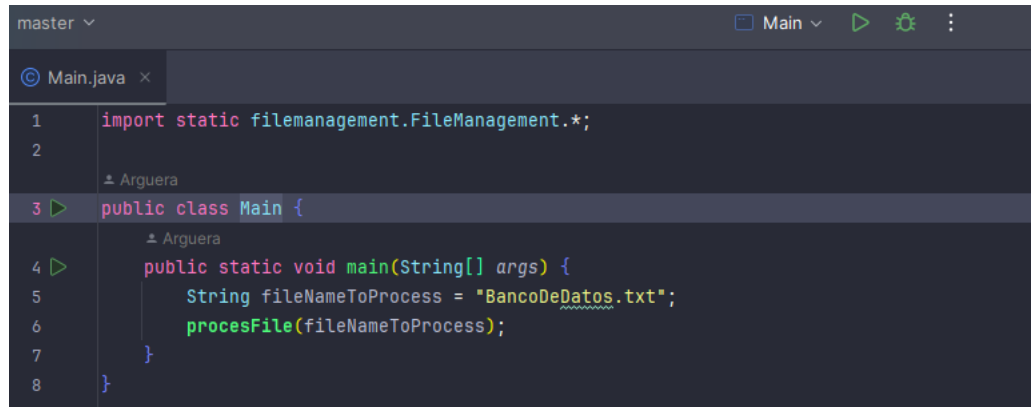
0 Issues

0 Stars

0 Forks

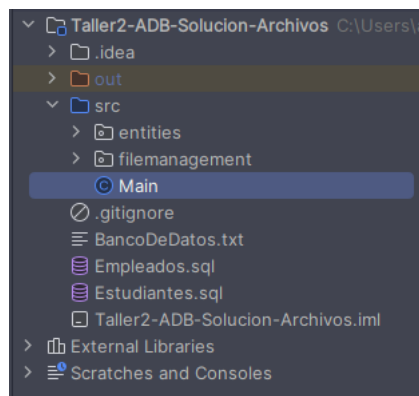
Una vez que ha sido clonado el repositorio, procedemos a abrirlo en el entorno de desarrollo IntelliJ, localizándonos en el archivo principal (main).





```
1 import static filemanagement.FileManagement.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         String fileNameToProcess = "BancoDeDatos.txt";
6         procesFile(fileNameToProcess);
7     }
8 }
```

Una vez nos encontramos en el archivo principal (main), procederemos a ejecutar el programa. Es importante destacar que se realizó una modificación en el nombre del archivo, pasando de "Banco De Datos.txt" a "BancoDeDatos.txt", con el fin de prevenir posibles conflictos derivados de espacios. La ejecución del código se lleva a cabo mediante el botón representado por un triángulo verde en la interfaz de IntelliJ IDEA. Este proceso generará archivos con extensión .sql, los cuales se encuentran listos para ser empleados en Oracle.



IMPORTANTE: Aquí se refleja la utilización del programa para el procesado del archivo de texto, opcionalmente puede solo ejecutar los archivos `Empleados.sql` y `Estudiantes.sql` los cuales contienen los datos listos para insertar en su instancia de base de datos.

Creación de usuarios

Para este ultimo punto de la guía, se deben crear usuarios para cada usuario registrado en la base de datos. A cada usuario se le debe asignar una contraseña de 10 caracteres. Además le asignaremos a cada usuario dependiendo su categoría un `TABSPACE` de los que ya definimos previamente. Una cuota de escritura ilimitada y por último la asignación de algún `ROL` ya definido.

Asignación de Usuarios a estudiantes

Para asignar usuarios a cada estudiante ya registrado en la base de datos, el equipo opto por realizar un proceso almacenado de Transact SQL.

Como primer paso el programador realizo la siguiente función que genera una contraseña aleatoria para cada usuario. Es importante ejecutar esta función primero ya que utilizaremos de esta mas adelante en el proceso de creación de usuarios, la función es la siguiente:

```
-- Funcion para crear una contraseña
CREATE OR REPLACE FUNCTION generar_password (p_longitud IN NUMBER) RETURN VARCHAR2 AS
    v_password VARCHAR2(100);
    v_caracteres CONSTANT VARCHAR2(95) := '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz#$_';
    v_letras CONSTANT VARCHAR2(52) := 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
    v_indice NUMBER;
BEGIN
    -- Generar el primer carácter como una letra
    v_indice := dbms_random.value(1, length(v_letras));
    v_password := v_password || SUBSTR(v_letras, v_indice, 1);
    -- Resto de la generación de la contraseña
    FOR i IN 2..p_longitud LOOP
        v_indice := dbms_random.value(1, length(v_caracteres));
        v_password := v_password || SUBSTR(v_caracteres, v_indice, 1);
    END LOOP;

    RETURN v_password;
END;
```

Ahora con la función de generación de contraseñas creada, procedemos a ejecutar el siguiente bloque anónimo de programación.

```
CREATE OR REPLACE PROCEDURE GENERATE_ESTUDIANTE_USERS IS
    password VARCHAR2(15);
    carnet VARCHAR2(20);
    user_name VARCHAR(20);
    --
    CURSOR CURSOR_ESTUDIANTE IS
        SELECT carnet
        FROM udbadmin.estudiante;
BEGIN
    OPEN CURSOR_ESTUDIANTE;
    LOOP FETCH CURSOR_ESTUDIANTE INTO carnet;
        user_name := ('U_' || carnet); --Nombre del usuario a crear
        password := (generar_password(10)); --Contraseña generada
        EXIT WHEN CURSOR_ESTUDIANTE%NOTFOUND;
        -- Agregar código para crear estudiante
        EXECUTE IMMEDIATE
        'CREATE USER ' || user_name ||
        ' IDENTIFIED BY ' || password ||
        ' DEFAULT TABLESPACE t_estudiantes ' ||
        ' TEMPORARY TABLESPACE temp ' ||
        ' QUOTA UNLIMITED ON t_estudiantes';
        DBMS_OUTPUT.PUT_LINE('Usuario: ' || user_name || ', Password: ' || password);
    END LOOP;
    CLOSE CURSOR_ESTUDIANTE;
END;
```

Luego de recibir la salida en consola de que el proceso PL/SQL ha sido exitoso, ejecutamos el proceso. Este recorrerá todos los usuarios asignándoles un Nombre de usuario y una contraseña, la cual podemos verificar en la salida DMBS para poder visualizarlos

```
-- Ejecutamos el proceso
EXEC GENERATE_ESTUDIANTE_USERS;
```

Asignación de Usuarios a Empleados

De la misma manera el equipo decidió utilizar otro proceso almacenado para generar los usuarios a los empleados, este de igual manera nos muestra el Nombre y Contraseña de los usuarios de los empleados.

El proceso almacenado es el siguiente

```
CREATE OR REPLACE PROCEDURE GENERATE_EMPLEADO_USERS IS
    password VARCHAR2(15);
    carnet VARCHAR2(20);
```



```

user_name VARCHAR2(20);
id_tipo_empleado INT;
CURSOR CURSOR_EMPLEADO IS
    SELECT carnet, id_tipo_empleado
    FROM udbadmin.empleado;
BEGIN
    OPEN CURSOR_EMPLEADO;
    LOOP
        FETCH CURSOR_EMPLEADO INTO carnet, id_tipo_empleado;
        EXIT WHEN CURSOR_EMPLEADO%NOTFOUND;
        user_name := 'U_' || carnet; -- Nombre del usuario a crear
        password := generar_password(10); -- Contraseña generada
        -- Crear usuario
        EXECUTE IMMEDIATE
            'CREATE USER ' || user_name ||
            ' IDENTIFIED BY ' || password ||
            ' DEFAULT TABLESPACE t_empleados ' ||
            ' TEMPORARY TABLESPACE temp';
        -- Asignar roles según el id_tipo_empleado
        CASE id_tipo_empleado
            WHEN 0 THEN
                EXECUTE IMMEDIATE 'GRANT administrativo TO ' || user_name;
            WHEN 1 THEN
                EXECUTE IMMEDIATE 'GRANT coordinador TO ' || user_name;
            WHEN 2 THEN
                EXECUTE IMMEDIATE 'GRANT docente TO ' || user_name;
            ELSE
                DBMS_OUTPUT.PUT_LINE('Usuario: ' || user_name || ', Password: ' || password || ', Valor no reconocido');
        END CASE;
        DBMS_OUTPUT.PUT_LINE('Usuario: ' || user_name || ', Password: ' || password || ' creado con éxito.');
```

Ahora ejecutamos el proceso

```
EXEC GENERATE_EMPLEADO_USERS;
```

Adicionalmente, usted puede verificar sus usuarios en la base de datos con la siguiente consulta:

```
SELECT * FROM dba_users
```

Cualquier duda o comentario. Puede realizarla a los respectivos autores de esta guía

Juan Neftaly Castellanos Hernández 00182222@uca.edu.sv
 Oscar Armando Calderón Arguera 00090822@uca.edu.sv