

INFORME – Entrega 1: Oracle + Conexión Java

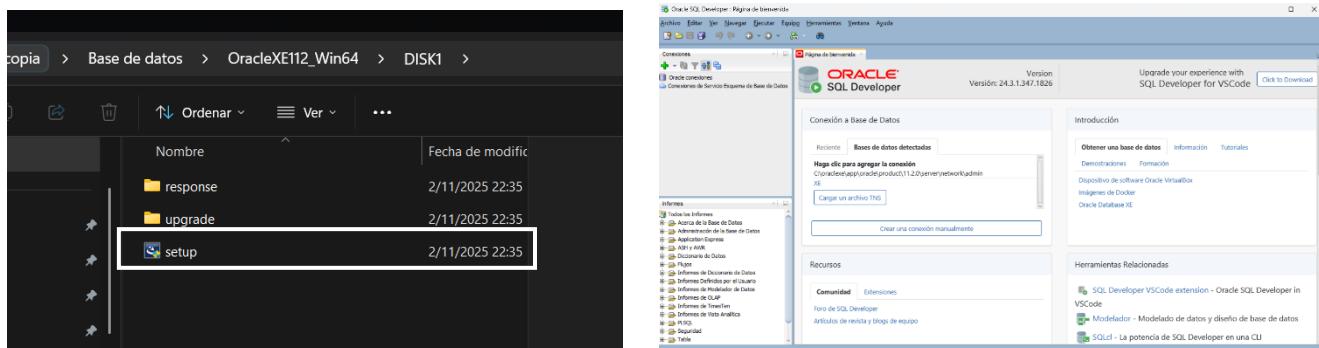
1. Instalación de Oracle 11g (pasos y configuración)

Se realizó la instalación de Oracle Database 11g Express Edition en Windows 10. Los pasos fueron los siguientes:

1. Se ingresó a la página oficial de Oracle <https://www.oracle.com/database/technologies/xe-prior-release-downloads.html> y se descargó Oracle 11g XE para Windows 64 bits.
2. Se ejecutó el archivo setup.exe como administrador.
3. Se aceptaron las opciones por defecto en la instalación.
4. Se configuró la contraseña del usuario administrador SYSTEM.

Parámetro	Valor configurado
Usuario admin	SYSTEM
Contraseña	Oracle123

5. Se verificó el estado de los servicios Oracle (OracleServiceXE y OracleXETNSListener) en ejecución.
6. Se probó el acceso mediante SQL*Plus



2. Conexión a Oracle usando SQL*Plus

Para comprobar la instalación se realizó una conexión desde CMD:

```
Administrator: Símbolo del sistema - sqlplus system/Oracle123@localhost:1521/XE
Microsoft Windows [Versión 10.0.26100.6899]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>sqlplus system/Oracle123@localhost:1521/XE

SQL*Plus: Release 11.2.0.2.0 Production on Dom Nov 2 23:35:49 2025

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

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL>
```

3. Tema del Proyecto

Sistema de Gestión Hospitalaria con Múltiples Áreas

Objetivo: Administrar información clínica básica dentro de un hospital para facilitar la gestión de pacientes, médicos, citas y consultas médicas.

Módulos considerados:

- Registro de Pacientes
- Gestión de Médicos
- Especialidades
- Citas médicas
- Consultas médicas

4. Diseño del esquema básico (Diagrama Entidad–Relación)

El sistema de gestión hospitalaria se diseñó considerando cinco entidades principales: **PACIENTE, MÉDICO, ÁREA, CITA y HISTORIAL_CLÍNICO**.

Cada una representa un componente esencial del funcionamiento interno del hospital.

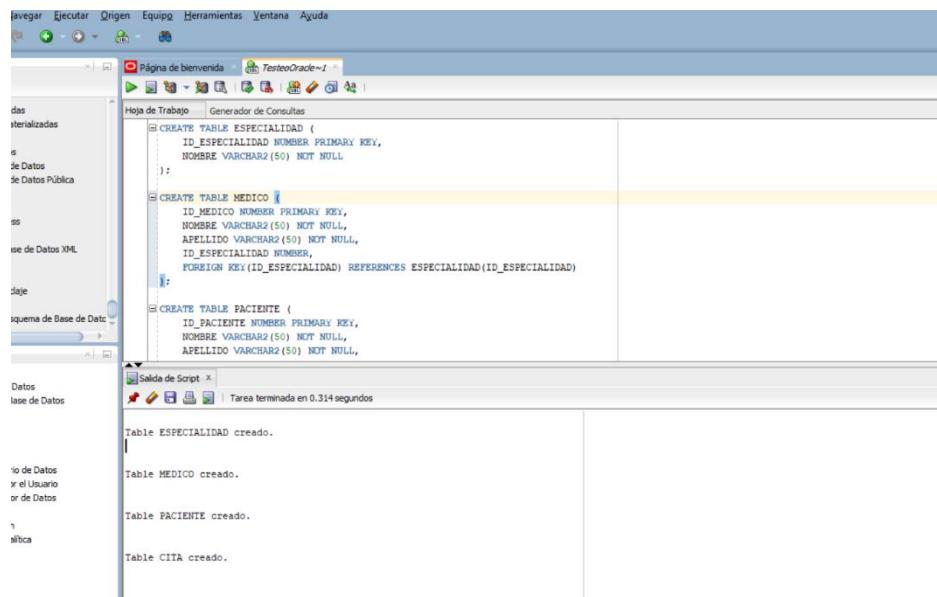
Relaciones definidas:

- Un **área** puede tener **muchos médicos** (Relación 1:N).
- Un **médico** puede atender **muchas citas** con pacientes (1:N).
- Un **paciente** puede tener **muchas citas** (1:N).
- Un **paciente** puede tener **muchos registros** en su historial clínico (1:N).



5. Creación de Tablas (Script SQL)

```
CREATE TABLE AREA (
    ID_AREA NUMBER PRIMARY KEY,
    NOMBRE VARCHAR2(50) NOT NULL
);
CREATE TABLE MEDICO (
    ID_MEDICO NUMBER PRIMARY KEY,
    NOMBRE VARCHAR2(50),
    APELLIDO VARCHAR2(50),
    ESPECIALIDAD VARCHAR2(50),
    TELEFONO VARCHAR2(15),
    ID_AREA NUMBER,
    CONSTRAINT FK_MEDICO_AREA FOREIGN KEY (ID_AREA)
        REFERENCES AREA(ID_AREA)
);
CREATE TABLE PACIENTE (
    ID_PACIENTE NUMBER PRIMARY KEY,
    NOMBRE VARCHAR2(50),
    APELLIDO VARCHAR2(50),
    DNI VARCHAR2(8),
    TELEFONO VARCHAR2(15),
    DIRECCION VARCHAR2(100)
);
CREATE TABLE CITA (
    ID_CITA NUMBER PRIMARY KEY,
    FECHA_CITA DATE,
    MOTIVO VARCHAR2(100),
    ID_PACIENTE NUMBER,
    ID_MEDICO NUMBER,
    CONSTRAINT FK_CITA_PACIENTE FOREIGN KEY (ID_PACIENTE)
        REFERENCES PACIENTE(ID_PACIENTE),
    CONSTRAINT FK_CITA_MEDICO FOREIGN KEY (ID_MEDICO)
        REFERENCES MEDICO(ID_MEDICO)
);
CREATE TABLE HISTORIAL_CLINICO (
    ID_HISTORIAL NUMBER PRIMARY KEY,
    DIAGNOSTICO VARCHAR2(200),
    TRATAMIENTO VARCHAR2(200),
    ID_PACIENTE NUMBER,
    CONSTRAINT FK_HISTORIAL_PACIENTE FOREIGN KEY (ID_PACIENTE)
        REFERENCES PACIENTE(ID_PACIENTE)
);
COMMIT;
```



6. Conexión desde Java (JDBC)

Se implementó una conexión básica entre Java y Oracle Database utilizando JDBC.

Se utilizó la clase `DriverManager` para establecer la conexión con el servicio `XE` ejecutado en localhost (127.0.0.1) en el puerto 1521.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConexionOracle {

    public static void main(String[] args) {

        // Datos de conexión
        String url = "jdbc:oracle:thin:@localhost:1521:XE";
        String user = "system";
        String password = "Oracle123"; // según tu configuración

        Connection conn = null;

        try {
            // Cargar el driver JDBC de Oracle
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Driver cargado correctamente ✅");

            // Conectar a Oracle
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("Conexión exitosa a Oracle ✅");

            // Crear consulta
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM
PACIENTE");

            // Mostrar los datos de la tabla PACIENTE
            System.out.println("\n❖ LISTA DE PACIENTES:");
            System.out.println("-----");
            while (rs.next()) {
                System.out.println("ID: " + rs.getInt("ID_PACIENTE") +
                    ", Nombre: " + rs.getString("NOMBRE") +
                    ", Apellido: " + rs.getString("APELIDO"));
            }

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                if (conn != null) conn.close();
                System.out.println("\nConexión cerrada ✅");
            } catch (Exception ex) {
                System.out.println("Error al cerrar: " +
ex.getMessage());
            }
        }
    }
}
```

The screenshot shows a terminal window with the following output:

```
C:\Users\dmill\jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program Files\Java\lib\jolokia.jar=port=8000" -jar BD_SQL_oracle.jar
Driver cargado correctamente ✓
Conexión exitosa a Oracle ✓

✖ LISTA DE PACIENTES:
-----
ID: 1, Nombre: Carlos, Apellido: Sánchez
ID: 2, Nombre: Ana, Apellido: Torres
ID: 3, Nombre: Pedro, Apellido: Castro

Conexión cerrada ✓

Process finished with exit code 0
```

7. Repositorio GitHub

Conclusiones

Durante el desarrollo del proyecto se logró instalar y configurar correctamente la base de datos Oracle 11g XE, asegurando su ejecución mediante el servicio local. Además, se crearon las tablas necesarias para la gestión académica y se insertaron registros haciendo uso de SQL Developer.

También se implementó una aplicación en Java utilizando el controlador JDBC, estableciendo la comunicación con Oracle, ejecutando operaciones CRUD (crear, leer, actualizar y eliminar) y validando la integridad de los datos.

Finalmente, se comprobó la integración entre la base de datos y la aplicación, lo cual refuerza el aprendizaje práctico del uso de SQL, modelado relacional y conexión cliente-servidor en un entorno real.