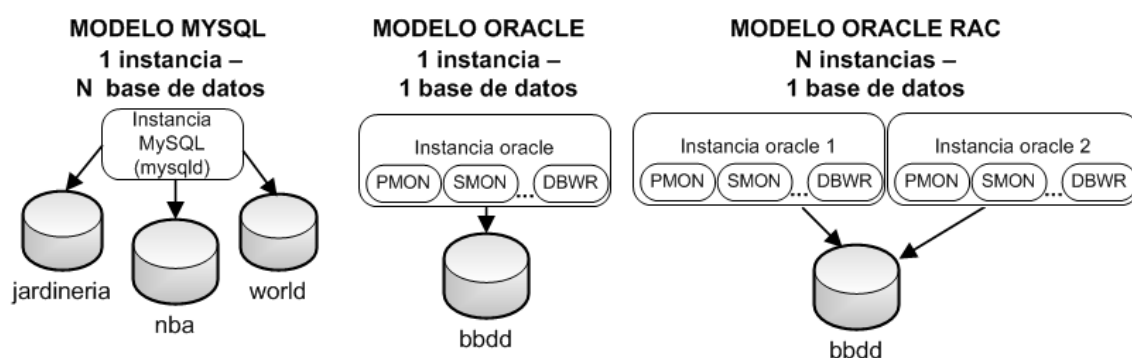


Java y Oracle

Introducción

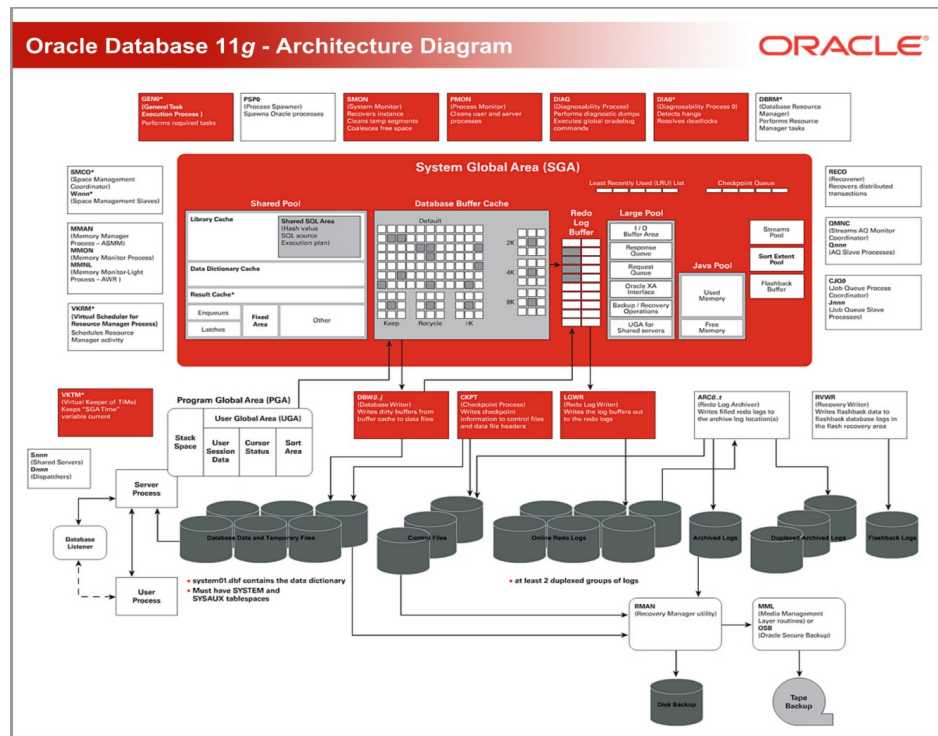
Antes de comenzar con el uso de Bases de Datos con Java, es importante conocer la diferencia entre instancia y base de datos. Una instancia es el conjunto de procesos del SGBD que están en ejecución y que permiten manipular una base de datos. En otros SGBD, una instancia puede manejar varias bases de datos, en el caso de Oracle una instancia solo maneja una única base de datos. Es más, en las instalaciones tipo RAC (Real Application Cluster), una base de datos puede ser manejada por varias instancias.



Por tanto, en Oracle, para poder crear una base de datos, primero se debe crear una instancia y después la BD. Esto se puede hacer de dos maneras: automáticamente a través de un asistente o mediante la línea de comandos.

Estructura de una instancia

Una instancia en Oracle se compone de varios procesos, cada uno con una función específica. Hay procesos clientes, como por ejemplo sqlplus que acceden a los datos, procesos servidores, creados para dar servicio a los procesos clientes y procesos en background que controlan el estado de la instancia.



Configuración de la red: El listener

Tras la creación de la base de datos, hay que ejecutar el comando "netca" (Net configuration Address) desde un terminal UNIX para configurar un **listener**, es decir, **un proceso servidor que escucha a través de los protocolos tcp-ip o ipc las peticiones de los clientes**. Normalmente se selecciona el puerto 1521. En el caso de Windows hay que ejecutar la orden "netstat -a" desde un terminal para comprobar que está escuchando.

```
C:\Users\María>netstat -a
```

Conexiones activas

Proto	Dirección local	Dirección remota	Estado
TCP	0.0.0.0:135	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:445	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:1521	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:5040	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:7680	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:8080	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:49664	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:49665	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:49666	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:49667	MARIA-LAPTOP:0	LISTENING
TCP	0.0.0.0:49668	MARIA-LAPTOP:0	LISTENING

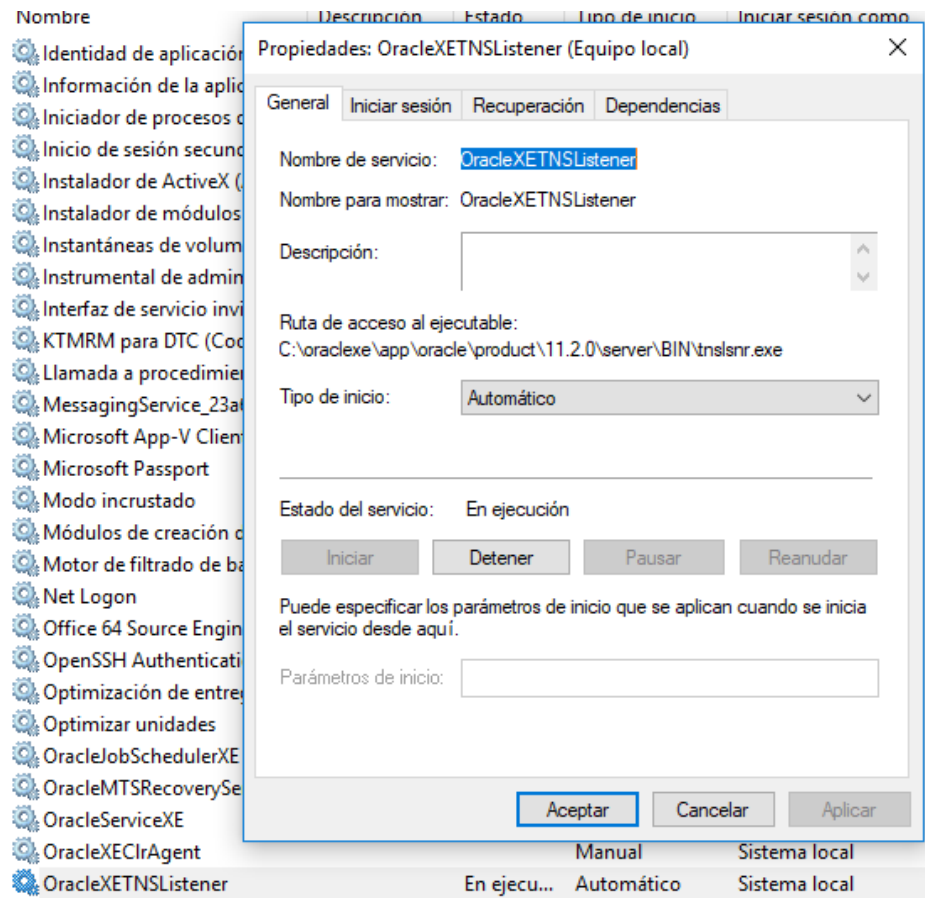
El fichero listener.ora que se encuentra en

<< oracle > product > 11.2.0 > server > network > ADMIN				Buscar en ADMIN
Nombre		Fecha de modifica...	Tipo	
sample		03/05/2019 17:45	Carpeta de archivos	
listener.ora		03/05/2019 17:45	Archivo ORA	
sqlnet.ora		29/05/2014 12:14	Archivo ORA	
tnsnames.ora		03/05/2019 17:45	Archivo ORA	

Este fichero declara un proceso escuchador que acepta conexiones TCP.

```
SID_LIST_LISTENER =  
  (SID_LIST =  
    (SID_DESC =  
      (SID_NAME = PLSExtProc)  
      (ORACLE_HOME = C:\oracle\app\oracle\product\11.2.0\server)  
      (PROGRAM = extproc)  
    )  
    (SID_DESC =  
      (SID_NAME = CLRExtProc)  
      (ORACLE_HOME = C:\oracle\app\oracle\product\11.2.0\server)  
      (PROGRAM = extproc)  
    )  
  )  
  
LISTENER =  
  (DESCRIPTION_LIST =  
    (DESCRIPTION =  
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1))  
      (ADDRESS = (PROTOCOL = TCP) (HOST = MARIA-LAPTOP) (PORT = 1521))  
    )  
  )  
  
DEFAULT_SERVICE_LISTENER = (XE)
```

Hay que tener en cuenta que cuando la base de datos se inicia, busca un listener libre y se registra automáticamente en él. De este modo, el listener conoce la existencia de la base de datos y puede así, despachar las solicitudes de acceso dirigidas a esta base de datos. Por tanto, para que todo funcione, tras un reinicio de máquina, debe arrancarse primero el listener y a continuación iniciar la instancia. Si el listener no está arrancado al iniciar la instancia, no escuchará peticiones para esa base de datos. En Windows hay que comprobar que están en ejecución los siguientes servicios de Windows: OracleServiceXE (instancia) y OracleXETNSlistener.



El fichero **tnsnames** sirve para definir direcciones de bases de datos y establecer conexiones con ellas a través del listener. Está ubicado en el mismo directorio que listener.ora

```
XE =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP) (HOST = MARIA-LAPTOP) (PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = XE)
  )
)
```

En host también se puede incluir localhost.

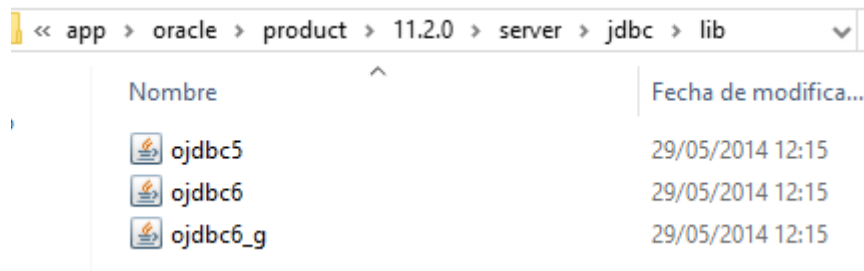
Java y Oracle




Oracle Database XE es una base de datos relacional que puede utilizar para almacenar, usar y modificar datos. Las aplicaciones Java utilizan el estándar de conectividad de base de datos Java (JDBC) para acceder y manipular datos en bases de datos relacionales.

JDBC es una interfaz de programación de aplicaciones (API) estándar en la industria que le permite acceder a una base de datos utilizando SQL desde Java.

Usando Java para conectarse a Oracle Database XE

- JDBC es un protocolo de acceso a la base de datos que le permite conectarse a una base de datos y ejecutar declaraciones y consultas SQL en la base de datos. Las bibliotecas de clases principales de Java proporcionan las API de JDBC, `java.sql` y `javax.sql`
- Para utilizar esta librería es necesario importarla al proyecto donde se estén utilizando las bases de datos.



Nombre	Fecha de modifica...
 <code>ojdbc5</code>	29/05/2014 12:15
 <code>ojdbc6</code>	29/05/2014 12:15
 <code>ojdbc6_g</code>	29/05/2014 12:15

- Para conectarse a una BD hay que seguir los siguientes pasos:
 1. Crear un driver de conexión con Oracle (<https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html>)

2. Establecer la conexión indicando

Driver :thin

Nombre de host: Nombre del host donde Oracle Database XE está instalado.

Si la base de datos está en el mismo ordenador el host es localhost .

Puerto JDBC :1521

SID: XE

Para conectarme a la base de datos Jardineria tendría que incluir

```
DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());

Connection conn = DriverManager.getConnection
    ("jdbc:oracle:thin:@localhost:1521:XE", "jardineria", "jardineria");
// driver@machineName:port:SID , userid, password
```

- Para ejecutar una consulta después de conectarnos a la base de datos hay que:
 1. Definir las sentencias de SQL con el método **createStatement** se utiliza para definir una declaración de consulta SQL.
 2. El método **executeQuery** se utiliza para ejecutar consultas en la base de datos y generar un conjunto de filas que coinciden con las condiciones de la consulta. Estos resultados están contenidos en un objeto **ResultSet**.
 3. Utiliza un objeto **ResultSet** para mostrar los datos

El siguiente ejemplo establece una conexión con Oracle y muestra por pantalla el código y el nombre de los clientes de jardinería.

```
import java.sql.*;

class dbAccess {
    public static void main (String args []) throws SQLException
    {

        DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());

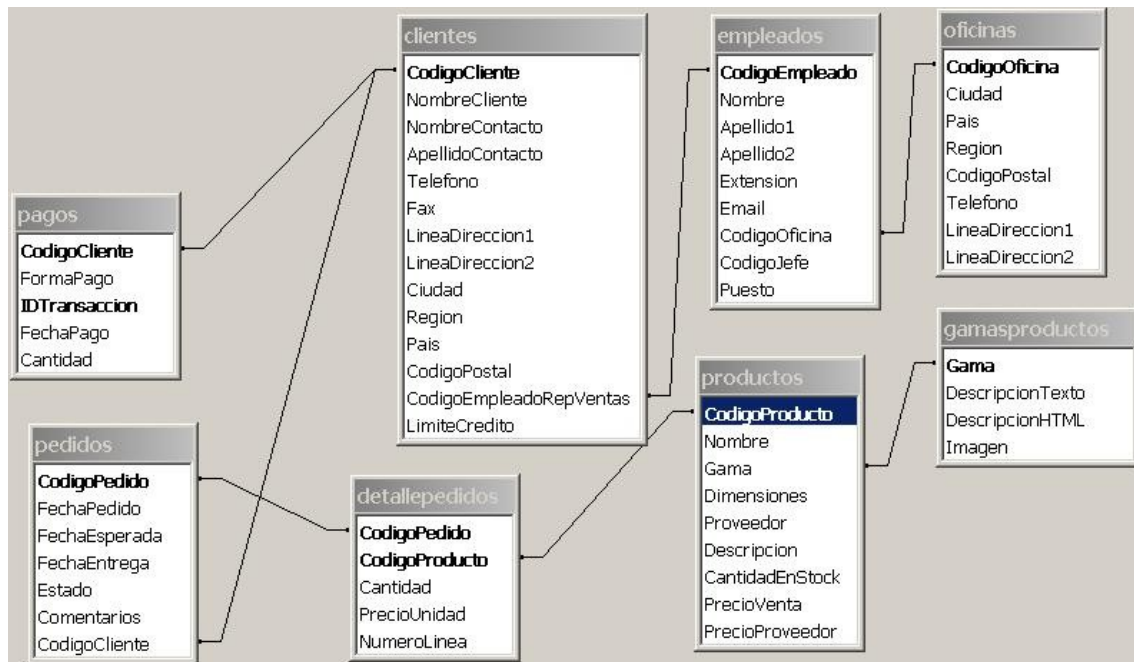
        Connection conn = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:XE", "jardineria", "jardineria");
            // driver@machineName:port:SID , userid, password

        Statement stmt = conn.createStatement();
        ResultSet rset =
            stmt.executeQuery("select CodigoCliente,NombreCliente from clientes");
        while (rset.next())
            System.out.println ("Codigo: "+rset.getString(1)+"||"+ "Nombre: "+rset.getString(2));
        stmt.close();

    }
}
```

Actividades

Para hacer los ejercicios de esta unidad se va a utilizar la BD jardinería. Para lo que hay que crear una base de datos llamada Jardineria con usuario y contraseña jardinería.



A continuación, crea un programa Java que:

1. Muestre por pantalla el código y el nombre de los clientes
2. La ciudad y el teléfono de las oficinas de EEUU
3. El nombre, los apellidos y el email de los empleados a cargo de Alberto Soria.
4. Nombre de clientes españoles
5. Crea una página web que tenga en una tabla el código, nombre, apellidos y teléfono de los clientes de la empresa

Crea un programa Java con el siguiente menú:

1. Crear pedidos.csv que crea un fichero pedidos.csv con código del pedido, el estado del pedido y el código del cliente. (Recuerda que los campos van separados por ;).
2. Crear detalle.csv: Crea otro llamado detalle.csv con todos los datos del detalle del pedido.
3. Cargar productos: Crea una clase llamada productos.java que tenga tantos campos como la tabla productos (getters, setters, constructores, toString).
Crea otro programa Java que realice una consulta de los productos y crea un arraylist de productos.

4. Consultar pedidos: Finalmente, crea un fichero llamado pedidos.txt donde muestres los productos que se han pedido y su detalle.

Referencias bibliográficas:

Bases de datos (2ª Edición): Iván López Montalbán y Manuel de Castro

Administración de Sistemas Gestores de bases de datos (2ª Edición): Iván López Montalbán

[https://docs.oracle.com/cd/E17781_01/appdev.112/e18805/
introduction.htm#TDPJD102](https://docs.oracle.com/cd/E17781_01/appdev.112/e18805/introduction.htm#TDPJD102)