

Asignatura

Gestión de Bases de datos

| Solucionario de ejercicios



UNIVERSAE



ÍNDICE

Unidad 1.....	3
Unidad 2.....	10
Unidad 3.....	16
Unidad 4.....	25
Unidad 5.....	42
Unidad 6.....	54
Unidad 7.....	61
Unidad 8.....	72

Unidad 1

Ejercicio 1.

Los ficheros utilizan diferentes terminología. Explica cuando se usa el termino Fichero o Archivo y su procedencia.

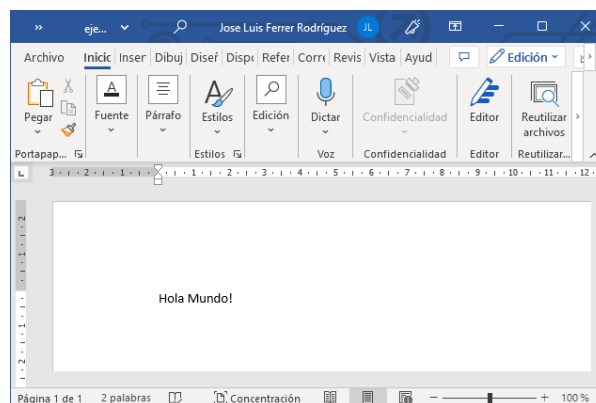
Ambos términos significan lo mismo, información almacenada en un dispositivo de almacenamiento. El término fichero a veces se utiliza para referirse a un documento de texto plano, mientras que el término archivo se puede utilizar para referirse a cualquier tipo de archivo, incluyendo imágenes, videos, etc.

Según su uso o procedencia podemos afirmar que fichero es más comúnmente utilizado en Europa, mientras que archivo es más comúnmente utilizado en América.

Ejercicio 2.

Crea un documento en Word y escribe la siguiente frase "Hola Mundo". Guárdalo y luego intenta abrirlo con el bloc de notas. ¿Es posible? ¿Se puede ver su contenido? ¿Qué tipo de fichero se trata según su contenido?

Es posible abrir el documento con el bloc de notas pero su contenido no es legible. Esto se debe a que se trata de un fichero con contenido binario y solo la aplicación que genero el fichero puede interpretar el contenido y mostrar Hola Mundo!, en cambio, cuando se abre con otra aplicación no es capaz de interpretar la información.







Ejercicio 4.

Tenemos la siguiente representación física de la información de vehículos.



Responde a las siguientes preguntas:

- ¿Qué tipo de acceso se utiliza?
- Identifica los punteros y los datos. Y explica para que sirven los punteros.
- Si estamos en el primer elemento "Coche" y queremos ir al elemento "Furgoneta" indica la secuencia de punteros que hay que recorrer.

El tipo de acceso que se muestra es secuencial encadenado.

Los punteros se encuentran al principio de cada elemento y son: 2, 4, 1 y 3
Los datos es la información que sigue al puntero y es posible identificar: "Coche", "Moto", "Furgoneta" y "Camión"

Los punteros contienen la dirección del siguiente elemento.

Si estamos en el primer elemento "Coche" identificamos el puntero a la izquierda con dirección 2, que indica que el siguiente elemento está en la posición 2.

A partir de aquí la secuencia para llegar al elemento Furgoneta es:

1. (2) Coche
2. (4) Moto
3. (3) Camión
4. (1) Furgoneta

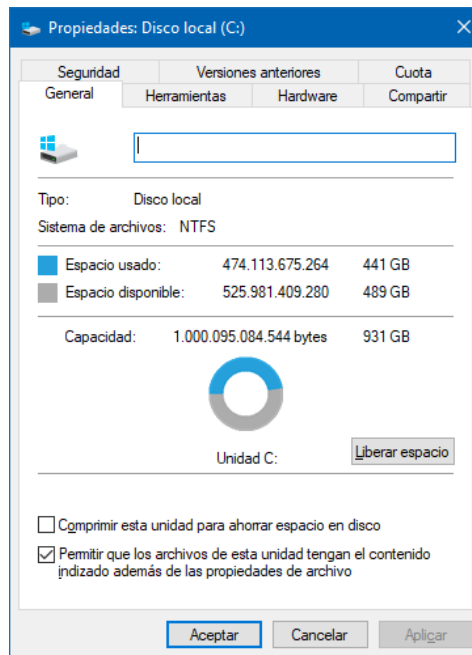
Ejercicio 5.

La gran mayoría de sistemas operativos dan diferentes opciones de acceso a sus ficheros. Realizar el siguiente procedimiento:

1. Accede a Mi Equipo
2. localiza la unidad C:
3. Pulsa botón secundario sobre la unidad y ve a Propiedades
4. En la pestaña General, mira si tienes activado la opción "Permitir que los archivos de esta unidad tengan el contenido indizado además de las propiedades de archivo"

Responde a las siguientes preguntas:

- ¿Qué realiza esta opción?
- ¿Qué beneficios aporta?
- ¿A qué tipo de acceso equivale?



Esta opción permite al sistema operativo comenzará a indexar el contenido de la unidad seleccionada, incluyendo el texto dentro de los archivos, no solo las propiedades de los archivos como el nombre y la fecha de creación. Esto puede ser útil para realizar búsquedas más precisas y rápidas de archivos específicos en la unidad, especialmente si la unidad contiene muchos archivos y carpetas.

Los beneficios que aporta:

- Búsqueda y acceso rápido a los archivos
- Precisión para encontrar los archivos concretos.
- Personalizar las búsquedas

El acceso que utiliza esta opción es: Indexado-encadenado.

Ejercicio 6.

A partir de la información siguiente:

CLIENTES

código – nombre – dirección – teléfono

001 – Frutocad S.A. – C/ Mariscal, 20, Madrid, 28001 – 600800900

002 – Hermanos Carrasco – C/ Las Flores, 120, Barcelona, 08001 – 800700900

003 – Cliclas S.L. – Av. España, 42, Barcelona, 08001 – 700800600

004 – TecnoPlusPlus S.L. – C/ Cantara, 13, Madrid, 28001 – 670600600

Realiza una tabla indicando que elementos se utilizan para representar la información para el tipo de base de datos relacional y orientado a objetos. Y pon un ejemplo a partir de la información que dispones de los Clientes.

Base de dato relacional	Orientado a objetos
Tabla o Relación	Clase

Fila, Tupla o Registro	Objeto o instancia
Columna o Campo	Atributo o campo

Ejemplo de representación de base de datos relacional.

- La tabla es todo el conjunto de Clientes
- Las columnas son: código, nombre, dirección y teléfono
- Las filas se pueden ver como cada línea con los datos en cada columna, por ejemplo 001, Frutocad S.A, C/ Mariscal, 20, Madrid, 28001, 600800900

Representación de la tabla completa

CODIGO	NOMBRE	DIRECCION	TELEFONO
001	Frutocad S.A	C/ Mariscal, 20, Madrid, 28001	600800900
002	Hermanos Carrasco	C/ Las Flores, 120, Barcelona, 08001	800700900
003	Cliclas S.L.	Av. España, 42, Barcelona, 08001	700800600
004	TecnoPlusPlus S.L.	C/ Cantara, 13, Madrid, 28001	670600600

Ejemplo de representación de los datos en las bases de datos orientado a objetos.

- La clase es la estructura de Clientes
- Los atributos o campos son: código, nombre, dirección y teléfono
- Los objetos se pueden ver como la materialización de la clase con los datos concretos, por ejemplo, new Cliente(001, "Frutocad S.A", "C/ Mariscal, 20, Madrid, 28001", 600800900)

```









Cliente {
    código: 001;
    nombre: "Frutocad S.A.";
    dirección: "C/ Mariscal, 20, Madrid, 28001";
    teléfono : 600800900
}

```

Ejercicio 7.

Investiga que SGDB son las más utilizadas actualmente, indica su tipo, si son comerciales o libres y explica las principales características de la que aparece en primer lugar.

Nota. La página web **DB-Engines** saca estadísticas de uso de software.

Rank			DBMS	Database Model	Score		
Apr 2023	Mar 2023	Apr 2022			Apr 2023	Mar 2023	Apr 2022
1.	1.	1.	Oracle +	Relational, Multi-model 	1228.28	-33.01	-26.54
2.	2.	2.	MySQL +	Relational, Multi-model 	1157.78	-25.00	-46.38
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model 	918.52	-3.49	-19.94
4.	4.	4.	PostgreSQL +	Relational, Multi-model 	608.41	-5.41	-6.05
5.	5.	5.	MongoDB +	Document, Multi-model 	441.90	-16.89	-41.48
6.	6.	6.	Redis +	Key-value, Multi-model 	173.55	+1.10	-4.05
7.	7.	8.	IBM Db2	Relational, Multi-model 	145.49	+2.57	-14.97
8.	8.	7.	Elasticsearch	Search engine, Multi-model 	141.08	+2.01	-19.76
9.	9.	10.	SQLite +	Relational	134.54	+0.72	+1.75
10.	10.	9.	Microsoft Access	Relational	131.37	-0.69	-11.41



Base de datos	Comercial / Libre	Tipo
Oracle	Comercial, pero ofrece una versión gratuita llamada Oracle Express Edition (XE)	Relacional / Objeto-relacional
MySQL	Libre	Relacional
Microsoft SQL Server	Comercial, pero Microsoft ofrece una versión gratuita llamada SQL Server Express	Relacional
PostgreSQL	Libre	Relacional
MongoDB	Libre, aunque MongoDB Inc. ofrece una versión comercial con características adicionales	NoSQL (Documental)
Redis	Libre	NoSQL (clave-valor)
Db2	Comercial	Relacional
Elasticsearch	Libre, aunque Elastic N.V. ofrece una versión comercial con características adicionales	NoSQL (Documental / Búsqueda y Análisis de textos)
SQLite	Libre	Relacional
Access	Comercial, aunque se incluye en algunas ediciones de Microsoft Office y se puede obtener como parte de una suscripción a Microsoft 365	Relacional

Características de Oracle

- Escalabilidad: Puede manejar grandes volúmenes de datos y usuarios simultáneos.
- Seguridad: Oracle ofrece autenticación de usuarios, control de acceso, auditoría y cifrado de datos.
- Fiabilidad: ofrece alta disponibilidad y fiabilidad, gracias a características como la recuperación ante desastres, la replicación y el clustering.
- Rendimiento: Es considerada de alto rendimiento que utiliza técnicas como el procesamiento en paralelo, la caché de resultados y la optimización de consultas para mejorar el rendimiento.
- Soporte para múltiples plataformas: Se ejecuta en una amplia gama de plataformas de hardware y sistemas operativos, lo que la hace altamente portátil.



- Gestión de datos: Dispone de herramientas para la gestión de datos, incluyendo copias de seguridad y restauración, migración de datos y herramientas de monitorización y análisis.
- Soporte para múltiples tipos de datos: Datos estructurados, semiestructurados y no estructurados.
- Arquitectura modular: Permite a los usuarios personalizar la base de datos para satisfacer sus necesidades específicas.

Requisitos de hardware

Procesador: al menos un procesador de cuatro núcleos de 64 bits.

Memoria RAM: al menos 8 GB de RAM, aunque se recomienda un mínimo de 16 GB para un rendimiento óptimo.

Espacio en disco: al menos 10 GB de espacio en disco duro para la instalación de Oracle Database, aunque se recomienda una cantidad mayor de espacio para almacenar datos y realizar copias de seguridad.



Unidad 2

Ejercicio1

Busca y explica todas las etapas que son necesarias para el proceso de diseño de una base de datos. ¿El diagrama E/R en que etapas se encuentra?

El proceso de diseño de una base de datos generalmente sigue las siguientes etapas:

1. **Requerimientos y análisis:** En esta etapa se recopila toda la información necesaria sobre el sistema que se quiere representar en la base de datos. Es importante entender cómo se utilizará la base de datos y qué tipo de información debe almacenar. Se analiza la información recopilada. Se identifican las entidades, atributos y relaciones que forman parte del sistema que se quiere representar en la base de datos. También se debe tener en cuenta quiénes serán los usuarios finales de la base de datos y cuáles serán sus necesidades.
2. **Diseño conceptual:** En esta etapa se crea un modelo conceptual de la base de datos utilizando un diagrama E/R (Entidad/Relación). Este modelo muestra las entidades y relaciones entre ellas, pero no entra en detalles sobre cómo se implementará físicamente la base de datos.
3. **Diseño lógico:** En esta etapa se transforma el modelo conceptual en un modelo lógico que puede ser implementado en un sistema de gestión de bases de datos (DBMS). Se define la estructura de la base de datos y se especifican los tipos de datos, las restricciones de integridad y las claves primarias y foráneas. En esta fase ya se tiene en cuenta el tipo de base de datos a utilizar.
4. **Diseño físico:** En esta etapa se define cómo se implementará físicamente la base de datos en el sistema de gestión de bases de datos (DBMS) seleccionado. Se especifican detalles como la distribución física de los datos en los dispositivos de almacenamiento, la definición de tablas, relaciones, índices y la optimización del rendimiento. Además se ven aspectos como el tamaño esperado de la base de datos, el número de usuarios concurrentes y el hardware disponible para alojar la base de datos.
5. **Implementación:** En esta etapa se crea la base de datos utilizando el modelo físico. Se crean las tablas, se definen las relaciones y se ingresan los datos.
6. **Pruebas:** En esta etapa se realizan pruebas para asegurarse de que la base de datos funciona correctamente. Se comprueba que los datos se puedan almacenar y recuperar correctamente y que se cumplan todas las restricciones de integridad.

7. Mantenimiento: En esta etapa se realiza el mantenimiento regular de la base de datos para asegurarse de que siga funcionando correctamente a medida que se van agregando más datos.

El diagrama E/R aparece en la etapa 2 Diseño conceptual.

Ejercicio 2.

En la siguiente imagen, identifica las principales entidades que aparecen, sus atributos, relaciones, cardinalidades, modalidades y justifica la decisión. Muestra un ejemplo de ocurrencia por cada entidad.



Entidades

- Restaurante (nombre, dirección)
- Camarero (dni, nombre, apellido, fechaContratacion, salario)
- Mesa (numeroMesa, ubicación, estado, numSillas)
- Producto (idProducto, nombre, categoría, precio)

Relaciones y cardinalidades:

- Un restaurante emplea varios camareros, y un camarero trabaja en un restaurante.
Cardinalidad: 1:N
Modalidad: Camarero (1,N) Restaurante (1, 1)
- Un restaurante tiene varias mesas, y cada mesa pertenece a un restaurante.
Cardinalidad: 1:N
Modalidad: Restaurante (1,1) Mesa (1,M)
- Un camarero atiende a varias mesas, y cada mesa es atendida por un camarero.
Cardinalidad: 1:N
Modalidad: Camarero (1,1) Mesa (0,N)

- En una mesa se puede servir varios productos, y cada producto puede ser servido para diferentes mesas.

Cardinalidad: M:N

Modalidad: Mesa (1,N) Producto (1,N)

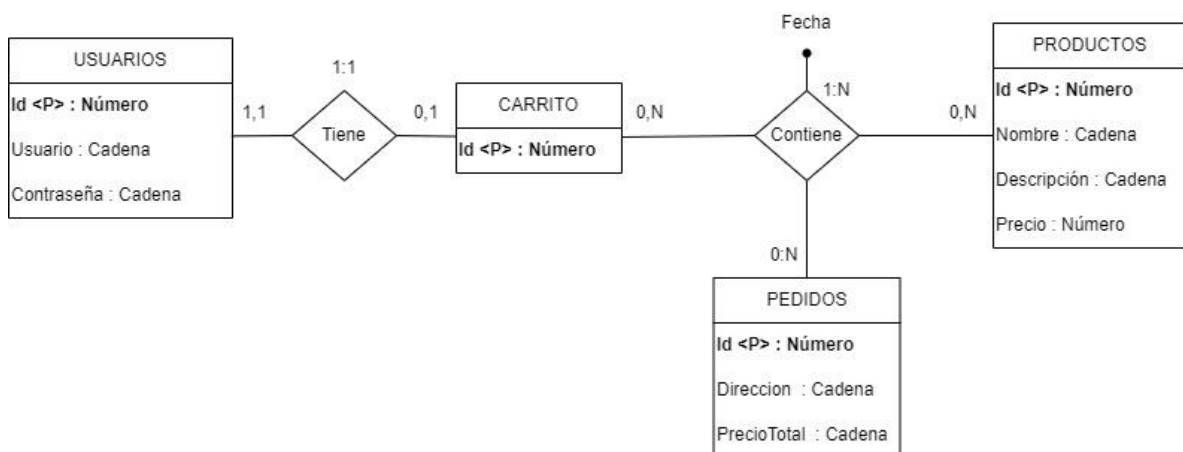
Ocurrencias

- Restaurante: "El Buen Sabor", "Calle Principal 123"
- Camarero: "11223344Y", "Juan", "Pérez", "15/01/2022", 1000
- Mesa: 3, "Exterior", "Ocupada", 4
- Producto: 12, "Cerveza", "Bebida", 3.50

Ejercicio 3.

Realiza el diagrama E/R y justifica las relaciones, cardinalidades y modalidades del siguiente enunciado:

"Una base de datos de comercio electrónico simple que permita a los usuarios navegar por productos, agregar artículos a un carrito de compras y realizar pedidos. La base de datos debe contemplar: usuarios, productos, carritos de compras y pedidos. Cada usuario debe tener un nombre de usuario y una contraseña, y cada producto debe tener un nombre, descripción y precio. El carrito de compras debe permitir a los usuarios agregar y eliminar artículos, y los pedidos debe contener información sobre los artículos pedidos, el coste total y la dirección de envío."



Relaciones y cardinalidades:

- Un usuario tiene un carrito, y cada carrito pertenece a un usuario. Es posible reflejar que existan usuario sin carritos, debido a que no han hecho nunca ningún pedido, y se consideren posibles clientes.
- Un usuario puede realizar varios pedidos, y cada pedido pertenece a un usuario. Como la base de datos tiene que reflejar un carrito, se considera la

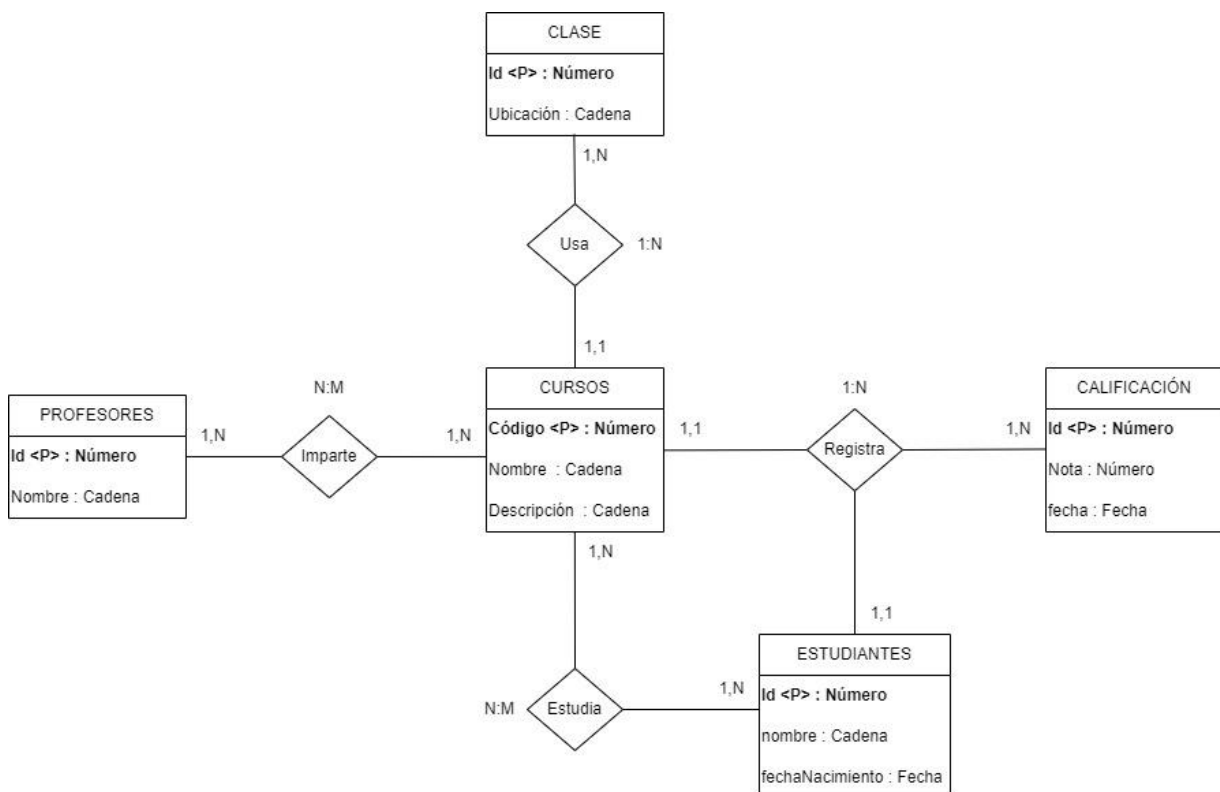
extensión del usuario, es posible que al tener un carrito aún no se haya realizado ningún pedido.

- Un carrito puede contener varios productos, y un producto puede estar en diferentes carritos. En esta relación surge un atributo de relación la fecha que permitirá conocer en qué fecha se puede formular el pedido que hay en el carrito.

Ejercicio 4.

Realiza el diagrama E/R y justifica las relaciones, cardinalidades y modalidades del siguiente enunciado:

"Una base de datos sobre el sistema escolar que realice un seguimiento de la información de los estudiantes y su rendimiento académico. La base de datos debe contemplar: estudiantes, cursos y calificaciones. Cada estudiante debe tener un nombre, fecha de nacimiento e identificación. Cada curso debe tener un código de curso, nombre y descripción, y cada calificación debe contener información sobre el rendimiento del estudiante en un curso en particular, como la calificación recibida y la fecha en que se obtuvo. La base de datos también debe incluir profesores y clases, con profesores asignados a cursos específicos y clases que contengan grupos de estudiantes matriculados en el mismo curso."



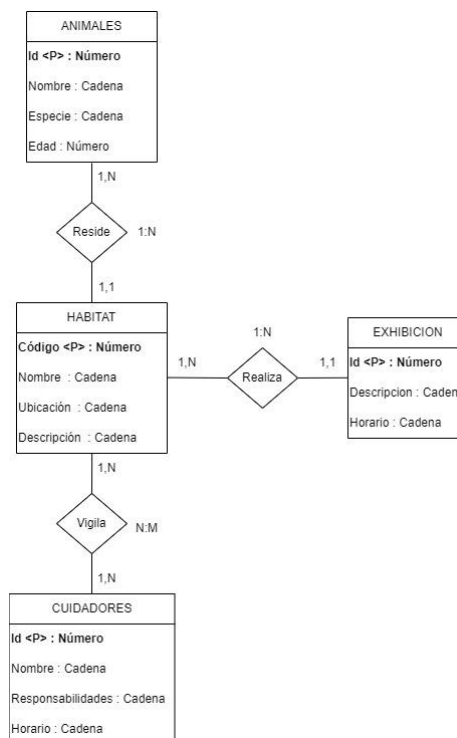
Relaciones y cardinalidades:

- Un estudiante puede estar matriculado en diferentes cursos y un curso debe tener múltiples estudiantes matriculados. Consideramos que un curso puede ser realizado mínimamente por un alumno.
- Un estudiante puede tener múltiples calificaciones (en diferentes cursos y momentos), y una calificación pertenece específicamente a un estudiante por curso.
- Un profesor puede impartir múltiples cursos y un curso puede ser impartido por diferentes profesores (por ejemplo, en diferentes semestres).
- Una clase está asociada a un único curso, pero un curso puede tener usar diferentes clases. Por ejemplo, un curso sanitaria, puede usar el aula teórica y un laboratorio.

Ejercicio 5.

Realiza el diagrama E/R y justifica las relaciones, cardinalidades y modalidades del siguiente enunciado:

"Un zoológico que incluya información sobre varios animales y sus hábitats. La base de datos debe contemplar: animales, hábitats y cuidadores. Cada animal debe tener un nombre, especie y edad, y cada hábitat debe tener un nombre, ubicación y descripción. El cuidador debe contener información sobre su nombre, sus responsabilidades y su horario. Cada animal debe ser asignado a un hábitat, y cada hábitat debe tener uno o más animales viviendo en él. La base de datos también debe incluir exhibiciones, donde cada exhibición contiene uno o más hábitats y proporciona información sobre los animales y sus hábitats a los visitantes."





Relaciones y cardinalidades:

- Un animal debe vivir en un único hábitat y un hábitat puede albergar uno o más animales.
- Un hábitat puede tener múltiples cuidadores (mínimo 1) y un cuidador puede vigilar un hábitat o más de uno.
- Una exhibición se puede realizar en uno o más hábitats, y un hábitat puede formar parte de una única exhibición.



Unidad 3

Ejercicio 1.

Existen más de una forma normal, en la práctica se suele trabajar hasta la 3FN. Para en otras formas normales investiga la Forma normal Boyce-Codd (FNBC) y responde a las siguientes preguntas:

- ¿Qué comprueba?
- ¿Qué requisitos tiene que cumplir para que se cumpla esta forma?

La Forma Normal de Boyce-Codd (BCNF o FNBC, por sus siglas en inglés) es una extensión de la Tercera Forma Normal (3FN) que aborda ciertos casos especiales en los que la 3FN puede permitir redundancias y anomalías. La BCNF se centra en las dependencias funcionales y en cómo estas afectan a la estructura de la base de datos.

¿Qué comprueba?

La BCNF comprueba que todas las dependencias funcionales en una tabla estén correctamente gestionadas, eliminando redundancias y anomalías que puedan ocurrir en ciertos casos, incluso si la tabla ya cumple con la 3FN. La BCNF garantiza que cada determinante (el atributo o conjunto de atributos que determina de manera única otro atributo) sea una clave candidata.

¿Qué requisitos tiene que cumplir para que se cumpla esta forma?

Para que una tabla esté en BCNF, debe cumplir con los siguientes requisitos:

- La tabla debe estar en 3FN.
- Para cada dependencia funcional no trivial $A \rightarrow B$ en la tabla, donde A y B son conjuntos de atributos, A debe ser una clave candidata. En otras palabras, si un atributo o conjunto de atributos determina de manera única otro atributo, ese determinante debe ser una clave candidata.

Si una tabla no cumple con la BCNF, se puede descomponer en múltiples tablas más pequeñas que sí cumplan con la BCNF, eliminando redundancias y anomalías. Sin embargo, es importante tener en cuenta que la BCNF es más restrictiva que la 3FN, y en algunos casos, puede dar lugar a una mayor cantidad de tablas y complejidad en la base de datos. Por lo tanto, se debe considerar cuidadosamente si es necesario aplicar la BCNF en función de las necesidades específicas del proyecto y los requisitos de rendimiento.

Ejercicio 2.

Pon un ejemplo por cada forma normal donde se pueda apreciar su violación:

- 1FN
- 2FN
- 3FN
- Forma normal Boyce-Codd (FNBC).

1FN

Código	Fecha de venta	Producto	Precio unitario	Cantidad
1	01/01/2022	1, 2, 3	10, 20, 30	1, 3, 2
2	02/01/2022	2, 3	20, 30	1, 4
3	03/01/2022	1, 2, 3, 4, 5	10, 20, 30, 40, 50	1, 2, 3, 4, 5

En esta tabla, la columna "Producto" contiene una lista de valores separados por comas, lo que viola la 1FN, ya que cada celda no contiene un valor atómico.

Para solucionarlo, se tendría que crear múltiples filas para registrar correctamente cada valor independiente en su campo.

Código	Fecha de venta	Producto	PrecioUnitario	Cantidad
1	01/01/2022	1	10	1
1	01/01/2022	2	20	3
1	01/01/2022	3	30	2
2	02/01/2022	2	20	1
2	02/01/2022	3	30	4
3	03/01/2022	1	10	1
3	03/01/2022	2	20	2
3	03/01/2022	3	30	3
3	03/01/2022	4	40	4
3	03/01/2022	5	50	5

2FN

<u>IdVenta</u>	Producto	Cantidad	PrecioUnitario	Categoría	Total
1	Bolígrafo	5	1.20	Material de oficina	6
2	Refresco	3	1.10	Bebidas	3.3
3	Peluche pequeño	7	0.90	Juguetes	6.3

La clave primaria es *IdVenta*

Esta tabla no cumple con la 2FN porque hay campos que no dependen íntegramente de la clave primaria. Existen algunas dependencias que no son exclusivas de la clave primaria, en concreto, la columna *Categoría* no depende íntegramente de *IdVenta*, si no de *Producto*. Además habría una dependencia parcial de *PrecioUnitario* con *Producto*.



Dependencias funcionales:

$\{IdVenta\} \rightarrow \{Producto\}$
 $\{IdVenta\} \rightarrow \{Cantidad\}$
 $\{IdVenta\} \rightarrow \{PrecioUnitario\}$
 $\{IdVenta\} \rightarrow \{Total\}$
 $\{Producto\} \rightarrow \{Categoria\}$
 $\{Producto\} \rightarrow \{PrecioUnitario\}$

Para solucionarlo, se quita la columna *PrecioUnitario* y *Categoria*. Y ubicamos estas columnas en otra tabla con la relación de producto.

<u>IdVenta</u>	idProducto	Cantidad	Total
1	100	5	6
2	200	3	3.3
3	300	7	6.3

<u>IdProducto</u>	Producto	PrecioUnitario	Categoria
100	Bolígrafo	1.20	Material de oficina
200	Refresco	1.10	Bebidas
300	Peluche pequeño	0.90	Juguetes

3FN

<u>idCurso</u>	<u>idClase</u>	idProfesor	nombreProfesor	Departamento
C1	S1	P1	José	Informática
C1	S2	P1	José	Informática
C2	S1	P2	Clara	Matemáticas
C3	S2	P1	José	Informática

La clave primaria es *idCurso* y *idClase*

Esta tabla no cumple con la 3FN. Todos los campos que no pertenecen a la clave primaria dependen íntegramente de la clave primaria. En cambio, entre los campos que no son clave primaria hay dependencia entre ellos. Se puede apreciar que *nombreProfesor* y *Departamento* dependen de *idProfesor*

Dependencias funcionales:

$\{IdCurso, idClase\} \rightarrow \{idProfesor\}$
 $\{IdCurso, idClase\} \rightarrow \{nombreProfesor\}$
 $\{IdCurso, idClase\} \rightarrow \{Departamento\}$
 $\{IdProfesor\} \rightarrow \{nombreProfesor\}$
 $\{IdProfesor\} \rightarrow \{Departamento\}$

Para solucionarlo, se moverán las columnas *nombreProfesor* y *Departamento* a otra tabla que lo relacione con Profesor.



<u>idCurso</u>	<u>idClase</u>	idProfesor
C1	S1	P1
C1	S2	P1
C2	S1	P2
C3	S2	P1

<u>idProfesor</u>	nombreProfesor	Departamento
P1	José	Informática
P2	Clara	Matemáticas

Forma normal Boyce-Codd(FNBC)

<u>idResponsable</u>	<u>idTrabajador</u>	DNIResponsable
1	1	11223344Y
1	2	11223344Y
2	2	22334477Z
3	3	99887766Q

La clave primaria es *idResponsable, idTrabajador*

Esta tabla no cumple la FNBC, Tenemos una relación para ver un listado de responsables y sus trabajadores. está en 3FN pero podemos ver que existen dos campos redundantes, *idResponsable* y *DNIResponsable*.

Dependencias funcionales:

{*idResponsable, idTrabajador*} → {*DNIResponsable*}
 {*DNIResponsable, idTrabajador*} → {*idResponsable*}

Para solucionarlo, hay dos posibilidades, establecer la clave primaria con *idResponsable* e *idTrabajador*, o *DNIResponsable* e *idTrabajador*. El campo sobrante se llevará a otra tabla.

<u>idResponsable</u>	<u>idTrabajador</u>
1	1
1	2
2	2
3	3

<u>idResponsable</u>	DNIResponsable
1	11223344Y
2	22334477Z
3	99887766Q

Ejercicio 3.

A partir de la siguiente tabla.

DNI	Nombre	Apellidos	CodigoCurso	NombreCurso
44556677G	Antonio	García Carrasco	001	Bases de datos
			002	Programación
88994455Y	Carolina	Martos Rodríguez	001	Bases de datos
22335577A	Eva	Pol Sánchez	002	Programación
			003	Interfaces

Donde la clave primaria es DNI.

Responde a las siguientes preguntas.

- Obtener las dependencias funcionales que existan.
- ¿Está en primera forma normal (1FN)? Justificarlo
- ¿Está en segunda forma normal (2FN)? Justificarlo
- ¿Está en tercera forma normal (3FN)? Justificarlo
- Has los cambios oportunos para que este en 3FN

a)

{DNI} → {Nombre}

{DNI} → {Apellidos}

{CodigoCurso} → {NombreCurso}

b)

No está en 1FN. Aparecen datos en los campos *CodigoCurso* y *NombreCurso* que se pueden dividir.

c)

Como no cumple la 1FN, tampoco está en 2FN.

d)

Como no se cumple la 1FN y la 2FN, tampoco está en 3FN.

e)

Para solucionarlo primero vamos a hacer que se cumpla la 1FN. Vamos a hacer que los valores de cada campo sean atómicos, se genera una fila por cada datos que aparece múltiple dentro de las columnas *CodigoCurso* y *NombreCurso*.

DNI	Nombre	Apellidos	CodigoCurso	NombreCurso
44556677G	Antonio	García Carrasco	001	Bases de datos
44556677G	Antonio	García Carrasco	002	Programación
88994455Y	Carolina	Martos Rodríguez	001	Bases de datos
22335577A	Eva	Pol Sánchez	002	Programación
22335577A	Eva	Pol Sánchez	003	Interfaces



Ahora aparece otro problema, no se cumple la regla de restricción de la clave primaria, aparecen registros repetidos solo para el campo DNI. Para solucionarlo, tenemos que modificar la clave primaria, agrupando los campos DNI y CodigoCurso.

<u>DNI</u>	<u>CodigoCurso</u>	Nombre	Apellidos	NombreCurso
44556677G	001	Antonio	García Carrasco	Bases de datos
44556677G	002	Antonio	García Carrasco	Programación
88994455Y	001	Carolina	Martos Rodríguez	Bases de datos
22335577A	002	Eva	Pol Sánchez	Programación
22335577A	003	Eva	Pol Sánchez	Interfaces

Como ya cumple la 1FN, ahora vamos a realizar las modificaciones oportunas para que cumpla la 2FN.

Con las dependencias que se habían obtenido en el apartado a) vemos que los campos que no forman parte de la clave primaria no dependen íntegramente de la clave primaria completa. *Nombre* y *Apellidos* solo dependen del campo *DNI* y *nombreCurso* solo dependen de *CodigoCurso*.

Generamos nuevas tablas que tengan solo las dependencias exclusivas que no forman parte de la clave primaria completa.

<u>DNI</u>	Nombre	Apellidos
44556677G	Antonio	García Carrasco
88994455Y	Carolina	Martos Rodríguez
22335577A	Eva	Pol Sánchez

<u>CodigoCurso</u>	NombreCurso
001	Bases de datos
002	Programación
003	Interfaces

<u>DNI</u>	<u>CodigoCurso</u>
44556677G	001
44556677G	002
88994455Y	001
22335577A	002
22335577A	003

Al separar en nuevas tablas ya se cumple la 2FN, ahora cada tabla tiene una clave primaria y los campos que no forman parte de la clave primaria, dependen exclusivamente de su clave primaria.

Como está en 2FN, y no hay más dependencias triviales de los campos que no forman parte de la clave primaria, podemos afirmar que está en 3FN.

Ejercicio 4.

Considera la siguiente tabla R con atributos A, B, C y D:

$R(A,B,C,D)$

donde A es la clave primaria. Las siguientes dependencias funcionales se cumplen:

$\{A\} \rightarrow \{BCD\}$

$\{BC\} \rightarrow \{D\}$

Responde a las siguientes preguntas

a) ¿Está R en primera forma normal (1FN)? Justificarlo.

b) ¿Está R en segunda forma normal (2FN)? Justificarlo.

c) ¿Está R en tercera forma normal (3FN)? Justificarlo.

d) ¿Está R en Boyce-Codd (FNBC)? Justificarlo.

e) Has los cambios oportunos para que este en FNBC

a)

Sí, R está en 1FN. La primera forma normal (1FN) requiere que todos los atributos de una tabla sean atómicos, es decir, no contengan conjuntos ni listas de valores. No hay información en el enunciado que indique que los atributos A, B, C o D contengan conjuntos o listas de valores, por lo que se puede asumir que la tabla R está en 1FN.

b)

Sí, R está en 2FN. La tabla está en 1FN y los atributos no clave dependen completamente de la clave primaria. En este caso, la clave primaria es {A} y las dependencias funcionales son $\{A\} \rightarrow \{BCD\}$. Todos los atributos no clave (B, C y D) dependen completamente de la clave primaria {A}, cumpliendo así la condición para 2FN.

c)

No, R no está en 3FN. Existen dependencias transitivas entre los atributos no clave. En este caso, existe una dependencia funcional $\{BC\} \rightarrow \{D\}$, que indica una dependencia entre atributos no clave. Por lo tanto, R no está en 3FN.

d)

No, R no está en FNBC. La forma normal de Boyce-Codd (FNBC) requiere que cada dependencia funcional $X \rightarrow Y$ en una tabla, X sea una superclave. En este caso, aunque la dependencia funcional $\{A\} \rightarrow \{BCD\}$ cumple con esta condición (ya que {A} es la clave primaria), la dependencia funcional $\{BC\} \rightarrow \{D\}$ no cumple con esta condición, ya que {BC} no es una superclave de la tabla R. Por lo tanto, R no está en FNBC.

e)

Para llevar la tabla R a FNBC, necesitamos descomponerla en dos tablas que no violen las condiciones de FNBC:

$R_1(A, B, C)$

Con clave primaria A

$R_2(B, C, D)$

Con clave primaria B y C

Ahora, en ambas tablas, la única dependencia funcional que se aplica es $\{A\} \rightarrow \{BC\}$ en R1 y $\{BC\} \rightarrow \{D\}$ en R2, cumpliendo con las condiciones de FNBC.

Ejercicio 5.

Considera la siguiente tabla R con atributos A, B, C, D y E:

$R(A,B,C,D,E)$

donde A es la clave primaria. Las siguientes dependencias funcionales se cumplen:

$\{A\} \rightarrow \{BC\}$

$\{B\} \rightarrow \{DE\}$

Responde a las siguientes preguntas

a) ¿Está R en primera forma normal (1FN)? Justificarlo.

b) ¿Está R en segunda forma normal (2FN)? Justificarlo.

c) ¿Está R en tercera forma normal (3FN)? Justificarlo.

d) ¿Está R en Boyce-Codd (FNBC)? Justificarlo.

e) Has los cambios oportunos para que este en FNBC

a)

Sí, R está en 1FN. No hay información en el enunciado que indique que los atributos A, B, C, D o E contengan conjuntos o listas de valores, por lo que se puede asumir que la tabla R está en 1FN.

b)

Sí, R está en 2FN. La segunda forma normal (2FN) requiere que una tabla esté en 1FN y que todos los atributos no clave dependan completamente de la clave primaria. En este caso, la clave primaria es $\{A\}$ y las dependencias funcionales son $\{A\} \rightarrow \{BC\}$. Todos los atributos no clave (B, C, D y E) dependen completamente de la clave primaria $\{A\}$, cumpliendo así la condición para 2FN.

c)

Sí, R está en 3FN. La tercera forma normal (3FN) requiere que una tabla esté en 2FN y que no existan dependencias transitivas entre los atributos no clave. Una dependencia transitiva ocurre cuando tienes una situación del tipo $A \rightarrow B$ y $B \rightarrow C$, lo que implica una dependencia transitiva $A \rightarrow C$. En este caso, no hay dependencias transitivas. B y C dependen de A, y D y E dependen de B.

d)

No, R no está en FNBC. La forma normal de Boyce-Codd (FNBC) requiere que cada dependencia funcional $X \rightarrow Y$ en una tabla, X sea una superclave. En este caso, aunque la dependencia funcional $\{A\} \rightarrow \{BC\}$ cumple con esta condición (ya que $\{A\}$ es la clave primaria), la dependencia funcional $\{B\} \rightarrow \{DE\}$ no cumple con esta condición, ya que $\{B\}$ no es una superclave de la tabla R. Por lo tanto, R no está en FNBC.



e)

Para llevar la tabla R a FNBC, necesitamos descomponerla en dos tablas que no violen las condiciones de FNBC:

R1(A, B, C)

Con clave primaria A

R2(B, D, E)

Con clave primaria B

Ahora, en ambas tablas, las dependencias funcionales que se aplican son $\{A\} \rightarrow \{BC\}$ en R1 y $\{B\} \rightarrow \{DE\}$ en R2, cumpliendo con las condiciones de FNBC.

Unidad 4

Ejercicio 1

Realiza la instalación de un servidor de base de datos relacional (MySQL o MariaDB) y contesta a las siguientes preguntas:

- Explica todo el proceso de instalación que has realizado
- Accede a la consola o terminal de la base de datos y procede a logearte. Ejecuta el siguiente comando: `select version()` y muestra una captura de pantalla.
- ¿Para qué sirve el usuario "root"?
- Si quisiéramos tener más un servidor de base de datos del mismo tipo activo en nuestro sistema operativo. ¿Qué necesitamos modificar para que no tengamos ningún problema?

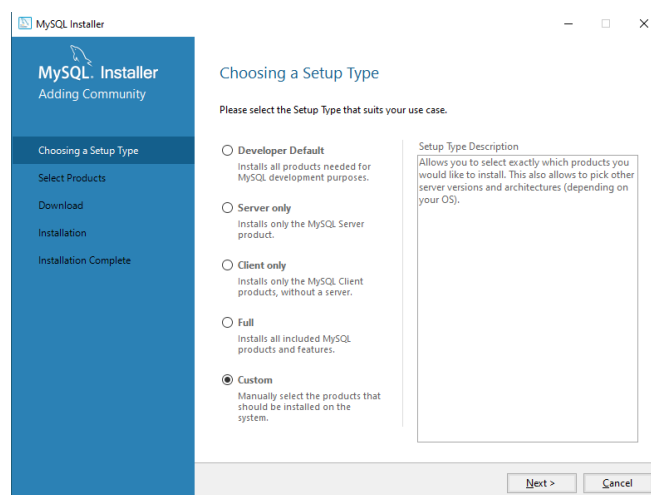
a)

Dependiendo del sistema operativo en el que se vaya a proceder la instalación el procedimiento de instalación puede cambiar.

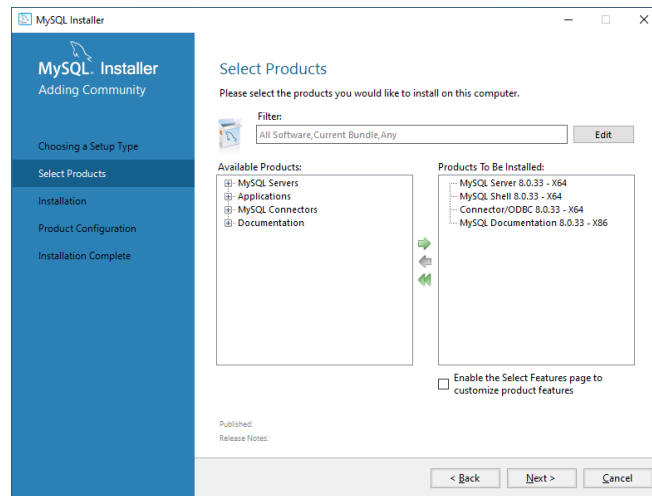
- Descargar el instalador de MySQL para el sistema operativo concreto de la página oficial. <https://dev.mysql.com/downloads/installer/>



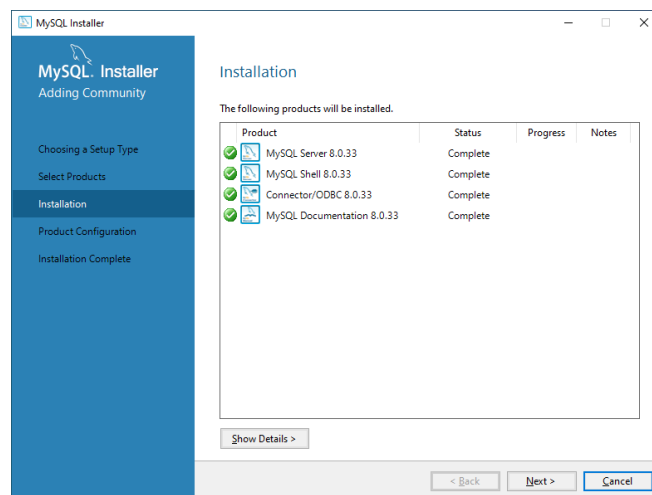
- Iniciamos el asistente de instalación. Y elegimos tipo de instalación "Custom"



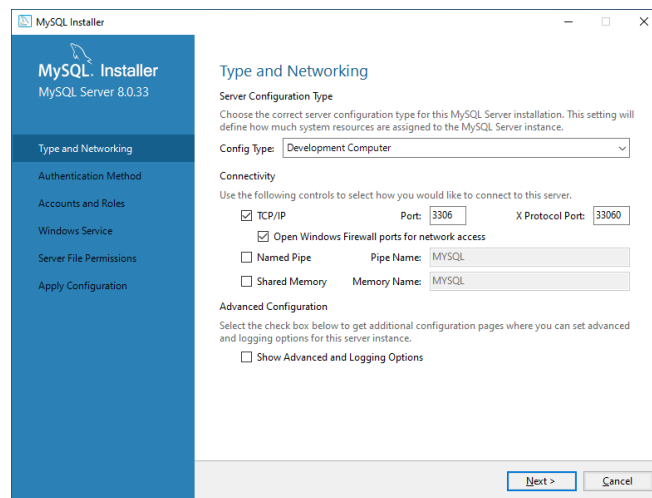
3. Elegimos que instale *MySQL Server X.X.XX*, *MySQL Shell X.X.XX*, *Connector/ODBC* y *MySQL Documentation*



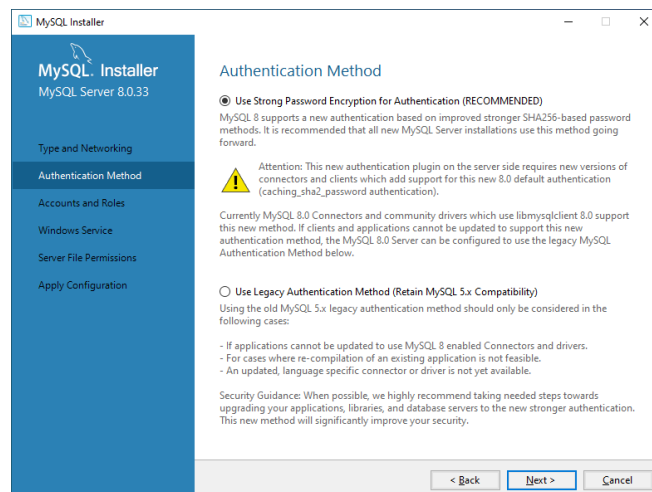
4. Se procede a ejecutar la instalación y verificar que completa correctamente cada aplicación.



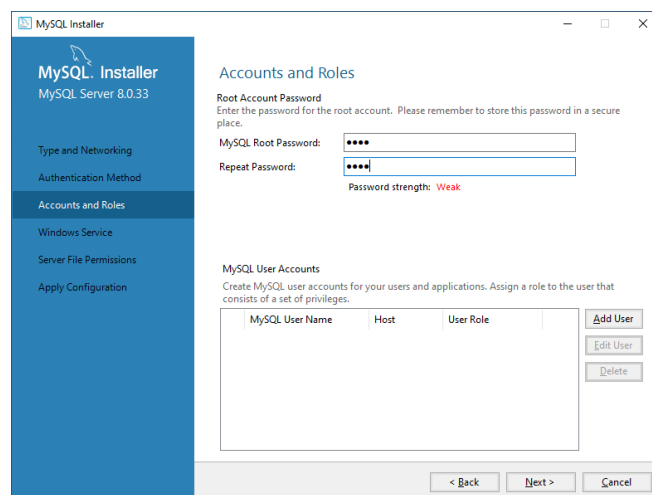
5. Configuramos el servidor MySQL. Dejamos todas las opciones por defecto. Es importante indicar que en este paso podemos cambiar el puerto por el que escucha el servidor.



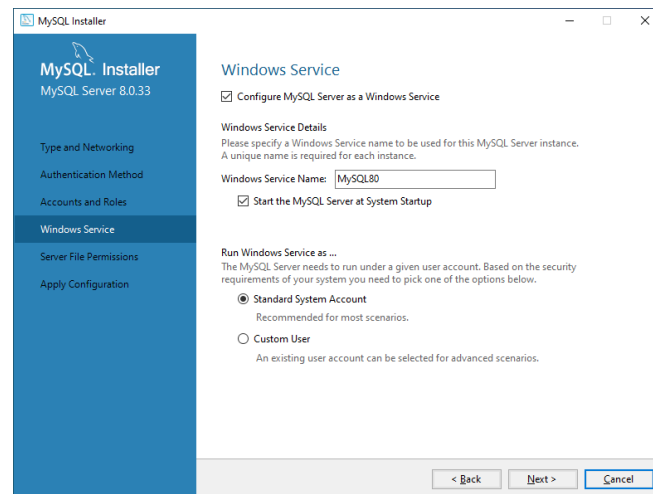
6. Especificamos el método de autenticación.



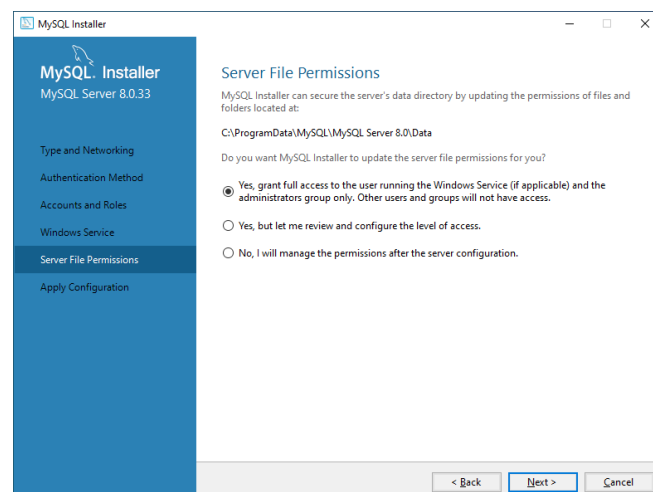
7. Este apartado es muy importante, establecemos la contraseña para el usuario "root". Además tenemos la posibilidad de añadir nuevos usuarios.



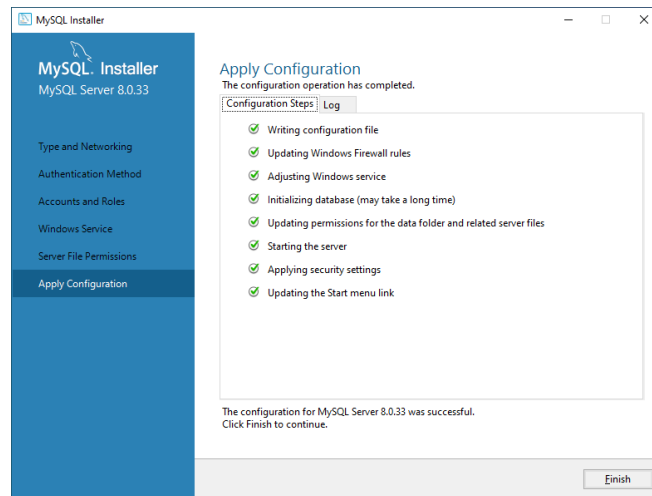
8. Configuramos el servicio del servidor en el sistema operativo Windows.



9. Indicamos los permisos que se otorgaran en el directorio de instalación.



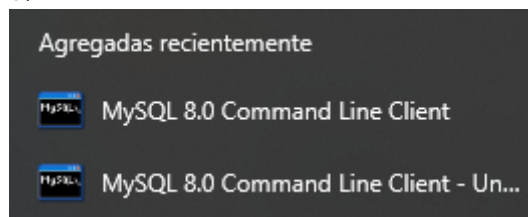
10. Ejecutamos la configuración. Si todo va bien aparecerá todo completado.



b)

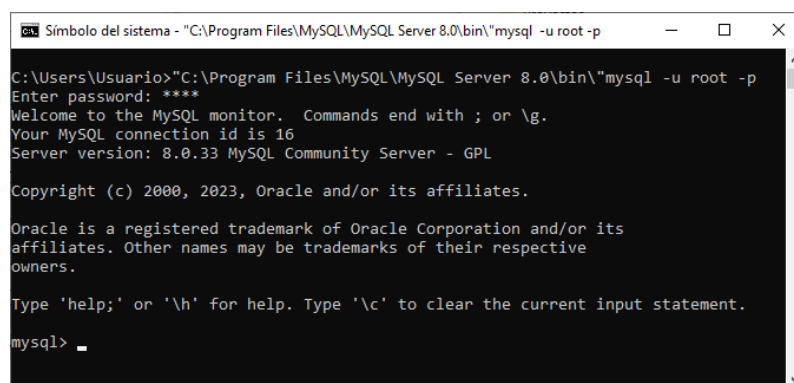
Para acceder al terminal de la consola:

1. Acceder directamente sobre el acceso directo que ha creado el proceso de instalación del paso a).



2. Iniciar una consola de Windows y ejecutar el siguiente comando:
"RUTA DE INSTALACION"\bin\mysql -u root -p

Se introduce la contraseña del usuario root





Ejecutamos el comando `select versión()`

```
Simbolo del sistema - "C:\Program Files\MySQL\MySQL Server 8.0\bin\"mysql -u root -p

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select version();
+-----+
| version() |
+-----+
| 8.0.33 |
+-----+
1 row in set (0.00 sec)

mysql>
```

c)

El usuario "root" es un usuario de gestión de la base de datos con privilegios completos. El propósito principal del usuario "root" es gestionar y administrar la base de datos y sus componentes, incluidas todas las bases de datos, tablas, usuarios y permisos.

d)

Para tener más de un servidor de base de datos tendríamos que hacer lo siguiente:

1. Indicar un directorio diferente de instalación.
2. Definir un puerto diferente a cada servidor. Si se utiliza el mismo habría un problema para iniciar cada servidor. Por defecto MySQL y MariaDB utilizan el puerto 3306.

Ejercicio 2


Realiza la instalación de un cliente de base de datos gráfico (SQLDeveloper, HeidiSQL, MySQL Workbench, Dbeaver, etc.).

- a) Explica todo el proceso de instalación que has realizado.
- b) Accede mediante el cliente y conéctate. Ejecuta el siguiente comando: `select versión()` y muestra una captura de pantalla.
- c) ¿Por qué es necesario un cliente para conectarse a la base de datos?
- d) Describe las ventajas e inconvenientes de utilizar un cliente de consola o terminal a uno gráfico.

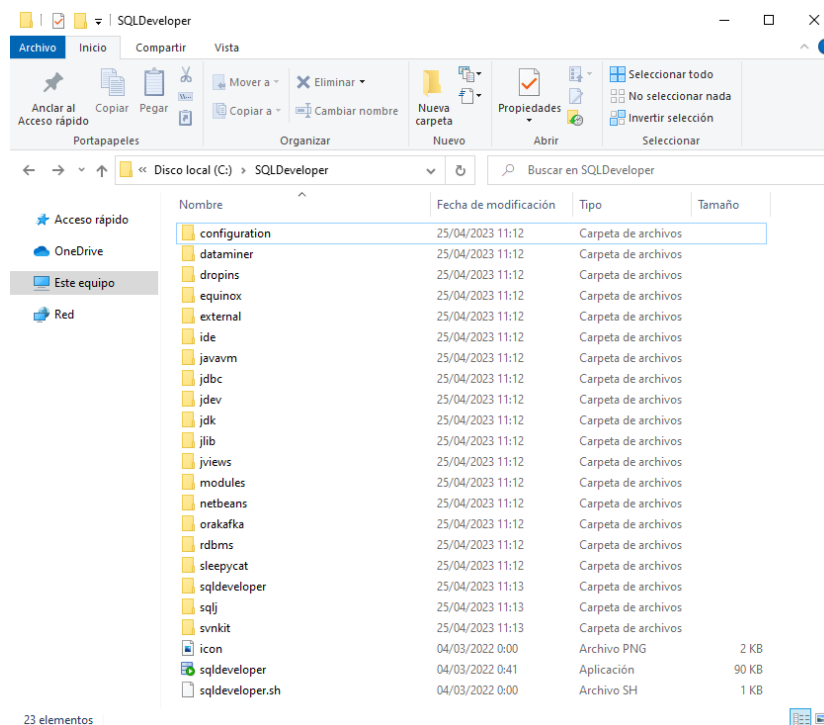
a)

Se procede a realizar la instalación de SQLDeveloper.

1. Descargamos de la página oficial el instalador.
<https://www.oracle.com/database/sqldeveloper/technologies/download/>
Elegiremos la opción que incluye la JDK de Java incorporada en la instalación. Si elegimos otra opción del cliente hay que realizar la instalación de la JDK de Java aparte.

Platform	Download	Notes
Windows 64-bit with JDK 11 included	 Download (459 MB)	<ul style="list-style-type: none"> MD5: 5b0fe5b6a5f53451204b26491da412f7 SHA1: 884f86c70130c109bb0cefd294eb8061ae1fd0ff Installation Notes

- Lo que hemos descargado es un archivo comprimido .zip. Lo descomprimos en una carpeta ubicada en C:\



- Abrimos el cliente desde el icono sqldeveloper
- Descargar el driver de conexión a MySQL, necesario para SQLDeveloper. Desde la página <https://downloads.mysql.com/archives/c-j/>

MySQL Connector/J (Archived Versions)

Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Connector/J, please visit [MySQL Downloads](#).

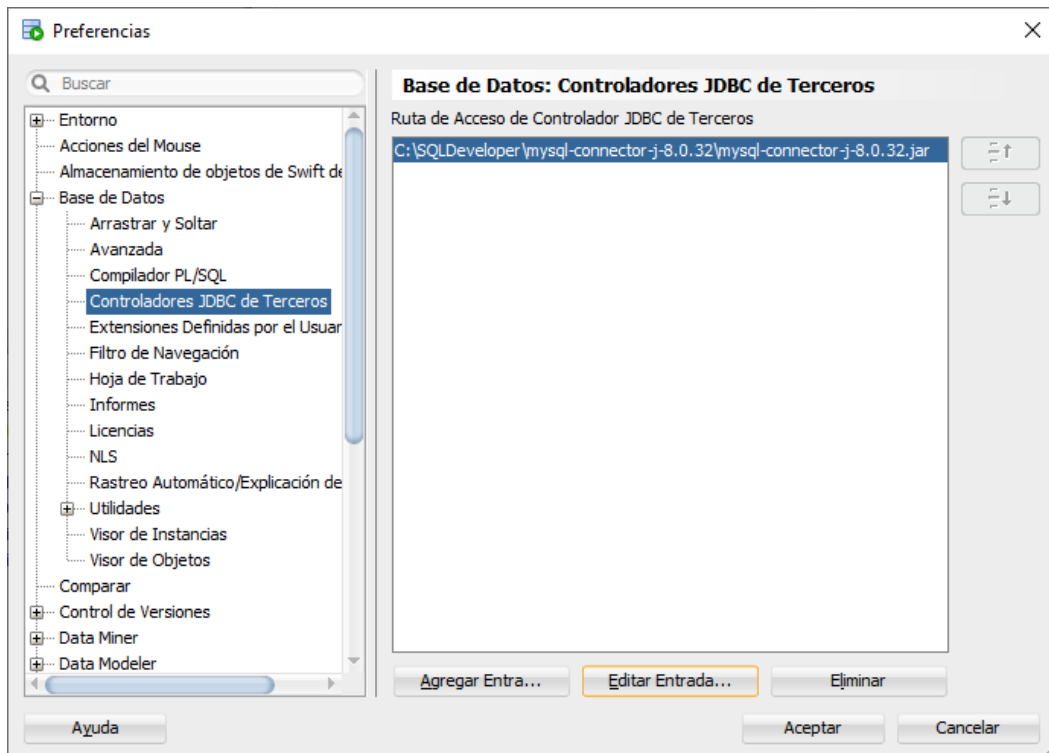
Product Version:

Operating System:

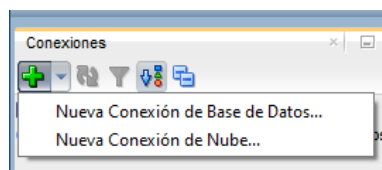
Platform Independent (Architecture Independent), Compressed TAR Archive	Dec 7, 2022	4.0M	Download
(mysql-connector-j-8.0.32.tar.gz)		MD5: 978655fc0f34e4d8d7ab591508b4944e Signature	
Platform Independent (Architecture Independent), ZIP Archive	Dec 7, 2022	4.8M	Download
(mysql-connector-j-8.0.32.zip)		MD5: 14bd829688fd6471a308c13f876bb2a4 Signature	

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

5. Extraemos el contenido dentro de la carpeta de instalación de SQLDeveloper
6. Desde SQLDeveloper en el menú Herramientas/Preferencias. Nos dirigimos a Base de Datos y Controladores JDBC de Terceros. En este apartado añadimos el driver mysql-connector-j-X.X.XX.jar que habíamos descomprimido en el apartado 5.

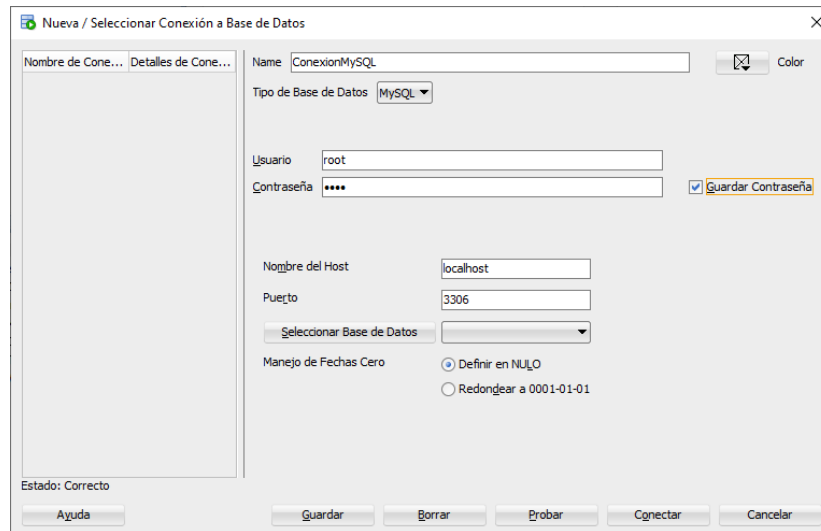


7. Volvemos a la pantalla inicial de SQLDeveloper. Vamos a añadir una nueva conexión a la base de datos. En el apartado Conexiones, pulsamos al icono verde + y le damos a la opción Nueva Conexión de Base de Datos...

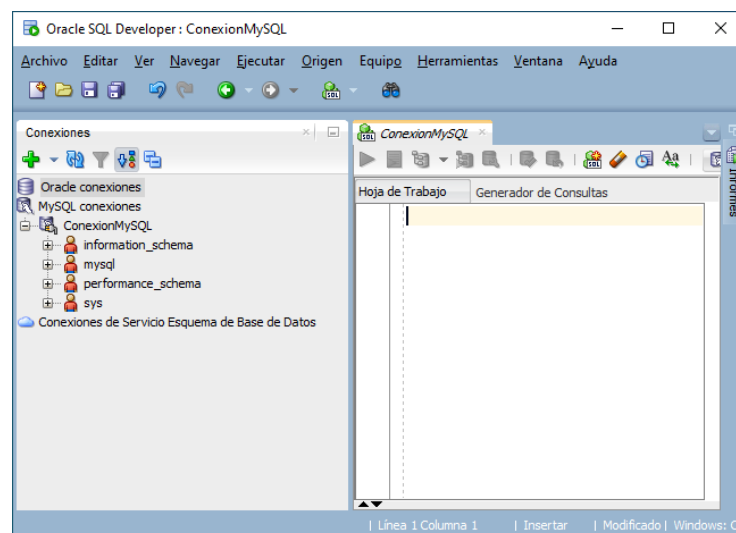


Indicamos un nombre a la conexión, y elegimos el tipo de base de datos MySQL. Como usuario y contraseña, se usará el usuario root y la contraseña que se puso en el proceso de instalación.

Por defecto, dejaremos el nombre del Host, localhost, si tenemos el servidor en otro equipo, se cambiará a la dirección específica. Si el puerto lo hemos modificado, se cambiará por el nuevo puerto.

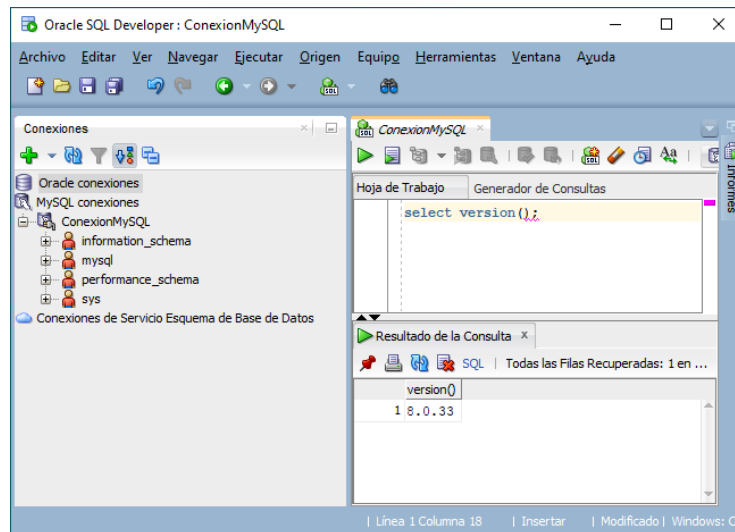


8. Una vez guardada la conexión, la tendremos disponible en el desplegable de conexiones. Al pulsar sobre la conexión, realizará la conexión y abrirá una hoja de trabajo para poder escribir cualquier instrucción. Para poder ejecutar cualquier instrucción se pulsará sobre el icono del play.



b)

Ejecución de la instrucción select version()



c)

La arquitectura de una base de datos sigue el esquema cliente – servidor.

En la arquitectura cliente-servidor, la base de datos y las aplicaciones de usuario se ejecutan en diferentes máquinas o procesos, y se comunican a través de una red o protocolos de comunicación específicos. Esta arquitectura permite separar las responsabilidades y tareas entre el cliente y el servidor, lo que conduce a una mejor organización, seguridad y escalabilidad.

El servidor se encarga de gestionar, almacenar y proteger los datos, así como de procesar las consultas y transacciones. Por otro lado, el cliente se encarga de interactuar con el usuario, enviar consultas al servidor y recibir y mostrar los resultados. Esta separación de roles hace que sea necesario un cliente de base de datos para conectarse al servidor y acceder a los datos almacenados en él.

d)

Los clientes de consola requieren menos recursos del equipo, suelen ser portátiles y ofrecen un mayor control y flexibilidad, pero tienen menos comodidades visuales y su uso requiere un nivel de dominio mayor.

Los clientes gráficos tienen interfaces de usuario intuitivas, funciones integradas y visualización de datos, pero pueden consumir más recursos y depender del software específico.

La elección entre un cliente de consola o gráfico dependerá de las necesidades, preferencias y habilidades de cada uno.

Ejercicio 3

A partir del siguiente modelo físico sobre animales y cuidadores procede a crear la estructura de la base de datos mediante sentencias DML.

ESPECIES (idEspecie, nombre, nombreCientifico, familia)
HABITATS (idHabitat, nombre, descripcion, capacidad)
ANIMALES (idAnimal, nombre, fechaNacimiento, genero, idEspecie, idHabitat)
 {idEspecie} es clave foránea de ESPECIES
 {idHabitat} es clave foránea de HABITATS
CUIDADORES (idCuidador, nombre, apellidos, email, teléfono)
 {email} es clave alternativa
ASIGNACIONES (idAnimal, idCuidador, fechaInicio, fechaFin)

```
CREATE TABLE Especies (  
    idEspecie INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(255) NOT NULL,  
    nombreCientifico VARCHAR(255) NOT NULL,  
    familia VARCHAR(255)  
);  
  
CREATE TABLE Habitats (  
    idHabitat INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(255) NOT NULL,  
    descripcion TEXT,  
    capacidad INT NOT NULL  
);  
  
CREATE TABLE Animales (  
    idAnimal INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(255) NOT NULL,  
    fechaNacimiento DATE,  
    genero ENUM('Masculino', 'Femenino') NOT NULL,  
    idEspecie INT,  
    idHabitat INT,  
    FOREIGN KEY (idEspecie) REFERENCES Especies(idEspecie),  
    FOREIGN KEY (idHabitat) REFERENCES Habitats(idHabitat)  
);  
  
CREATE TABLE Cuidadores (  
    idCuidador INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(255) NOT NULL,  
    apellidos VARCHAR(255) NOT NULL,  
    email VARCHAR(255) UNIQUE NOT NULL,  
    telefono VARCHAR(20) NOT NULL  
);
```



```
CREATE TABLE Asignaciones (  
    idAnimal INT NOT NULL,  
    idCuidador INT NOT NULL,  
    fechaInicio DATE NOT NULL,  
    fechaFin DATE,  
    PRIMARY KEY (idAnimal, idCuidador),  
    FOREIGN KEY (idAnimal) REFERENCES Animales(idAnimal),  
    FOREIGN KEY (idCuidador) REFERENCES Cuidadores(idCuidador)  
);
```

Ejercicio 4

A partir del siguiente modelo físico sobre películas e integrantes que participan en una película procede a crear la estructura de la base de datos mediante sentencias DML.

INTEGRANTES (idIntegrante, documentoIdentidad, **nombre**, **apellido**, **fechaNacimiento**, **nacionalidad**)
 {documentoIdentidad} es clave alternativa

ACTORES (idActor, tematicaPreferida, numeroOscars)
 {idActor} es clave foránea de INTEGRANTES

DIRECTORES (idDirector, numeroPeliculasFilmadas)
 {idDirector} es clave foránea de INTEGRANTES

PELICULAS (idPelicula, **titulo**, añoPublicacion, **idDirector**, duración)
 {idDirector} es clave foránea de DIRECTORES

GENEROS (idGenero, **nombre**, descripción)

PELICULA_ACTORES (idPelicula, idActor)
 {idPelicula} es clave foránea de PELICULAS
 {idActor} es clave foránea de ACTORES

PELICULA_GENEROS (idPelicula, idGenero)
 {idPelicula} es clave foránea de PELICULAS
 {idGenero} es clave foránea de GENEROS

Consideramos que se van a realizar muchas búsquedas por el campo nombre de los integrantes y por los nombres de los géneros.

```
CREATE TABLE INTEGRANTES (  
    idIntegrante INT PRIMARY KEY,  
    documentoIdentidad VARCHAR(255) UNIQUE,  
    nombre VARCHAR(255) NOT NULL,  
    apellido VARCHAR(255) NOT NULL,  
    fechaNacimiento DATE NOT NULL,  
    nacionalidad VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE ACTORES (  
    idActor INT PRIMARY KEY,  
    tematicaPreferida VARCHAR(255),  
    numeroOscars INT,  
    FOREIGN KEY (idActor) REFERENCES INTEGRANTES (idIntegrante)  
);  
  
CREATE TABLE DIRECTORES (  
    idDirector INT PRIMARY KEY,  
    numeroPeliculasFilmadas INT,  
    FOREIGN KEY (idDirector) REFERENCES INTEGRANTES (idIntegrante)  
);  
  
CREATE TABLE PELICULAS (  
    idPelicula INT PRIMARY KEY,  
    titulo VARCHAR(255) NOT NULL,  
    añoPublicacion INT,  
    idDirector INT NOT NULL,  
    duracion INT,  
    FOREIGN KEY (idDirector) REFERENCES DIRECTORES (idDirector)  
);  
  
CREATE TABLE GENEROS (  
    idGenero INT PRIMARY KEY,  
    nombre VARCHAR(255) NOT NULL,  
    descripcion TEXT  
);  
  
CREATE TABLE PELICULA_ACTORES (  
    idPelicula INT,  
    idActor INT,  
    PRIMARY KEY (idPelicula, idActor),  
    FOREIGN KEY (idPelicula) REFERENCES PELICULAS (idPelicula),  
    FOREIGN KEY (idActor) REFERENCES ACTORES (idActor)  
);  
  
CREATE TABLE PELICULA_GENEROS (  
    idPelicula INT,  
    idGenero INT,  
    PRIMARY KEY (idPelicula, idGenero),  
    FOREIGN KEY (idPelicula) REFERENCES PELICULAS (idPelicula),  
    FOREIGN KEY (idGenero) REFERENCES GENEROS (idGenero)  
);  
  
CREATE INDEX idx_integrantes_nombre ON INTEGRANTES (nombre);  
  
CREATE INDEX idx_generos_nombre ON GENEROS (nombre);
```

Ejercicio 5

Diseñar una base de datos para gestionar una biblioteca. La información que se necesita almacenar incluye datos sobre autores, libros, editores, categorías, copias de libros, usuarios y préstamos. A continuación, se describen los requerimientos y las relaciones entre las distintas entidades:

- Un autor tiene un identificador único, documento de identidad, nombre, apellido, fecha de nacimiento y nacionalidad. El documento de identidad es una clave alternativa.
- Un libro tiene un identificador único, título, año de publicación, identificador del autor y número de páginas. Cada libro está asociado a un autor.
- Un editor tiene un identificador único, nombre, dirección y teléfono.
- Un libro puede estar asociado a varios editores y un editor puede estar asociado a varios libros.
- Una categoría tiene un identificador único, nombre y descripción.
- Un libro puede pertenecer a varias categorías y una categoría puede contener varios libros.
- Una copia de un libro tiene un identificador único, identificador del libro, ubicación y estado. Cada copia está asociada a un libro.
- Un usuario tiene un identificador único, documento de identidad, nombre, apellido, fecha de nacimiento y nacionalidad. El documento de identidad es una clave alternativa.
- Un préstamo tiene un identificador único, identificador del usuario, identificador de la copia, fecha de préstamo y fecha de devolución. Cada préstamo está asociado a un usuario y a una copia del libro.

Realiza el modelo físico y la estructura de la base de datos mediante sentencias DML

```
AUTORES (idAutor, documentoIdentidad, nombre, apellido, fechaNacimiento,
nacionalidad)
```

```
{documentoIdentidad} es clave alternativa
```

```
LIBROS (idLibro, titulo, añoPublicacion, idAutor, numeroPaginas)
```

```
{idAutor} es clave foránea de AUTORES
```

```
EDITORES (idEditor, nombre, direccion, telefono)
```

```
LIBROS_EDITORES (idLibro, idEditor)
```

```
{idLibro} es clave foránea de LIBROS
```

```
{idEditor} es clave foránea de EDITORES
```

```
CATEGORIAS (idCategoría, nombre, descripcion)
```

```
LIBROS_CATEGORIAS (idLibro, idCategoría)
```

```
{idLibro} es clave foránea de LIBROS
```

```
{idCategoría} es clave foránea de CATEGORIAS
```



COPIAS (idCopia, idLibro, ubicacion, estado)
{idLibro} es clave foránea de LIBROS

USUARIOS (idUsuario, documentoIdentidad, nombre, apellido, fechaNacimiento, nacionalidad)
{documentoIdentidad} es clave alternativa

PRESTAMOS (idPrestamo, idUsuario, idCopia, fechaPrestamo, fechaDevolucion)
{idUsuario} es clave foránea de USUARIOS
{idCopia} es clave foránea de COPIAS

```
CREATE TABLE AUTORES (  
  idAutor INT PRIMARY KEY,  
  documentoIdentidad VARCHAR(255) UNIQUE,  
  nombre VARCHAR(255) NOT NULL,  
  apellido VARCHAR(255) NOT NULL,  
  fechaNacimiento DATE NOT NULL,  
  nacionalidad VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE LIBROS (  
  idLibro INT PRIMARY KEY,  
  titulo VARCHAR(255) NOT NULL,  
  añoPublicacion INT,  
  idAutor INT NOT NULL,  
  numeroPaginas INT,  
  FOREIGN KEY (idAutor) REFERENCES AUTORES (idAutor)  
);  
  
CREATE TABLE EDITORES (  
  idEditor INT PRIMARY KEY,  
  nombre VARCHAR(255) NOT NULL,  
  direccion VARCHAR(255),  
  telefono VARCHAR(255)  
);  
  
CREATE TABLE LIBROS_EDITORES (  
  idLibro INT,  
  idEditor INT,  
  PRIMARY KEY (idLibro, idEditor),  
  FOREIGN KEY (idLibro) REFERENCES LIBROS (idLibro),  
  FOREIGN KEY (idEditor) REFERENCES EDITORES (idEditor)  
);  
  
CREATE TABLE CATEGORIAS (  
  idCategoria INT PRIMARY KEY,  
  nombre VARCHAR(255) NOT NULL,  
  descripcion TEXT  
);
```

```
CREATE TABLE LIBROS_CATEGORIAS (  
    idLibro INT,  
    idCategoria INT,  
    PRIMARY KEY (idLibro, idCategoria),  
    FOREIGN KEY (idLibro) REFERENCES LIBROS (idLibro),  
    FOREIGN KEY (idCategoria) REFERENCES CATEGORIAS (idCategoria)  
);  
  
CREATE TABLE COPIAS (  
    idCopia INT PRIMARY KEY,  
    idLibro INT NOT NULL,  
    ubicacion VARCHAR(255) NOT NULL,  
    estado VARCHAR(255) NOT NULL,  
    FOREIGN KEY (idLibro) REFERENCES LIBROS (idLibro)  
);  
  
CREATE TABLE USUARIOS (  
    idUsuario INT PRIMARY KEY,  
    documentoIdentidad VARCHAR(255) UNIQUE,  
    nombre VARCHAR(255) NOT NULL,  
    apellido VARCHAR(255) NOT NULL,  
    fechaNacimiento DATE NOT NULL,  
    nacionalidad VARCHAR(255) NOT NULL  
);  
  
CREATE TABLE PRESTAMOS (  
    idPrestamo INT PRIMARY KEY,  
    idUsuario INT NOT NULL,  
    idCopia INT NOT NULL,  
    fechaPrestamo DATE NOT NULL,  
    fechaDevolucion DATE,  
    FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario),  
    FOREIGN KEY (idCopia) REFERENCES COPIAS (idCopia)  
);
```

Ejercicio de la videoclase

Queremos tener una base de datos que pueda reflejar la matriculación de alumnos en un centro. Elementos a tener en cuenta:

- ALUMNOS, CURSOS, DOCENTES, CENTROS.

Diseña una base de datos.

- Crea su base de datos
- Las tablas y campos que creáis oportunos
- Identificar y crear aquellos índices que sean necesarios.
- Establecer las relaciones que puedan existir.



```
CREATE DATABASE educacion;

USE educacion;

CREATE TABLE ALUMNOS (
    idAlumno INT PRIMARY KEY,
    documentoIdentidad VARCHAR(255) UNIQUE,
    nombre VARCHAR(255) NOT NULL,
    apellido VARCHAR(255) NOT NULL,
    fechaNacimiento DATE NOT NULL,
    direccion VARCHAR(255) NOT NULL
);

CREATE TABLE CURSOS (
    idCurso INT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    descripcion TEXT,
    duracion INT,
    idCentro INT NOT NULL,
    FOREIGN KEY (idCentro) REFERENCES CENTROS (idCentro)
);

CREATE TABLE DOCENTES (
    idDocente INT PRIMARY KEY,
    documentoIdentidad VARCHAR(255) UNIQUE,
    nombre VARCHAR(255) NOT NULL,
    apellido VARCHAR(255) NOT NULL,
    fechaNacimiento DATE NOT NULL,
    especialidad VARCHAR(255) NOT NULL
);

CREATE TABLE CENTROS (
    idCentro INT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    direccion VARCHAR(255) NOT NULL,
    telefono VARCHAR(255)
);

CREATE TABLE ALUMNOS_CURSOS (
    idAlumno INT,
    idCurso INT,
    PRIMARY KEY (idAlumno, idCurso),
    FOREIGN KEY (idAlumno) REFERENCES ALUMNOS (idAlumno),
    FOREIGN KEY (idCurso) REFERENCES CURSOS (idCurso)
);

CREATE TABLE DOCENTES_CURSOS (
    idDocente INT,
    idCurso INT,
    PRIMARY KEY (idDocente, idCurso),
    FOREIGN KEY (idDocente) REFERENCES DOCENTES (idDocente),
    FOREIGN KEY (idCurso) REFERENCES CURSOS (idCurso)
);
```

Unidad 5

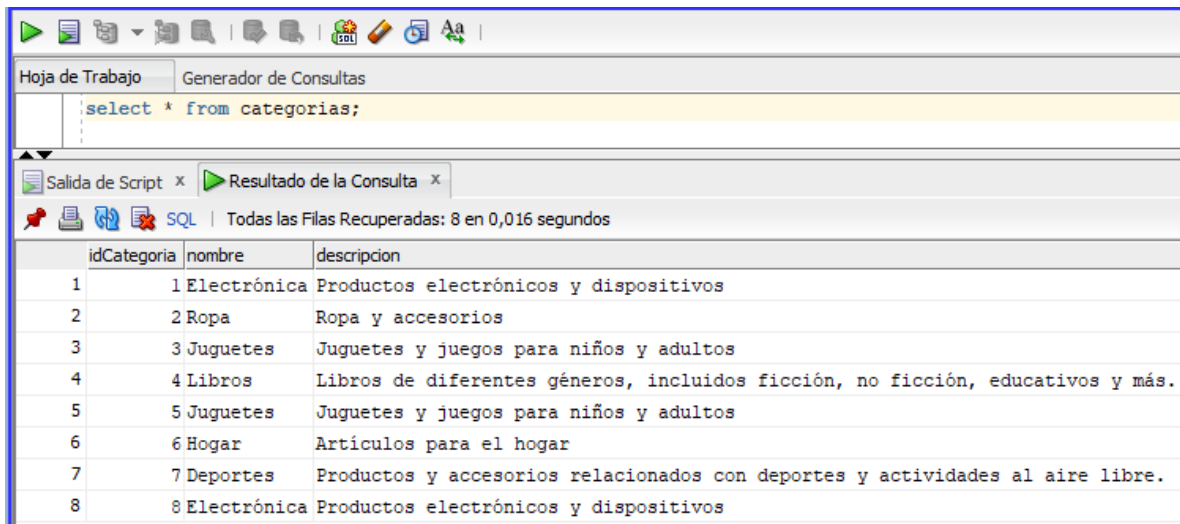
Ejercicio 1

Responde a las siguientes preguntas:

- Pon una captura donde se vean todas las categorías. ¿Existen categorías repetidas?
- Muestra todas las sentencias que son necesarias para que aparezca solo una categorías de cada tipo.
- Revisar si cada categoría tiene el identificador como muestra la tabla. Si no, realiza los cambios oportunos y muestra las sentencias que han sido necesarias.
- Revisar que los productos tienen la categoría correcta. Si no es así, realiza los cambios oportunos y muestra las sentencias que han sido necesarias para corregirlo.

Esta solución puede variar ligeramente dependiendo de la carga de datos que se haya realizado.

a)



The screenshot shows a SQL query editor with the following content:

```
select * from categorias;
```

The results are displayed in a table with the following columns: idCategoria, nombre, and descripcion. The table contains 8 rows of data, with some categories appearing to be duplicated (e.g., 'Electrónica' and 'Juguetes' appear twice each).

idCategoria	nombre	descripcion
1	1 Electrónica	Productos electrónicos y dispositivos
2	2 Ropa	Ropa y accesorios
3	3 Juguetes	Juguetes y juegos para niños y adultos
4	4 Libros	Libros de diferentes géneros, incluidos ficción, no ficción, educativos y más.
5	5 Juguetes	Juguetes y juegos para niños y adultos
6	6 Hogar	Artículos para el hogar
7	7 Deportes	Productos y accesorios relacionados con deportes y actividades al aire libre.
8	8 Electrónica	Productos electrónicos y dispositivos

Si, existen categorías duplicadas, como Electrónica y Juguetes

b)

Para solucionarlo vamos a eliminar el registro con id 5 "Juguetes" y el registro con id 8 "Electrónica".

Antes de borrar cualquier registro hay que ver si ese registro es usado en otra tabla, para esta base de datos, categorías se utiliza en la tabla productos. Buscamos inicialmente si existe algún registro que usa esas categorías.



```
SELECT * FROM productos WHERE idCategoria = 5;
SELECT * FROM productos WHERE idCategoria = 8;
```

Podemos afirmar que la categoría con id 5 "Juguetes" si esta utilizada en la tabla de productos. Entonces procedemos a cambiar de categoría del registro que aparece en la tabla productos antes de borrarla. Si no hiciéramos estos pasos no podríamos borrar el registro de categorías.

```
UPDATE productos SET idCategoria = 4 WHERE idCategoria = 5;
DELETE FROM categorias WHERE idCategoria = 5;
```

```
DELETE FROM CATEGORIAS WHERE idCategoria = 8;
```

Una vez finalizado los cambios, ya no tenemos categorías duplicadas.

idCategoria	nombre	descripcion
1	1 Electrónica	Productos electrónicos y dispositivos
2	2 Ropa	Ropa y accesorios
3	3 Juguetes	Juguetes y juegos para niños y adultos
4	4 Libros	Libros de diferentes géneros, incluidos ficción, no ficción, educativos y más.
5	6 Hogar	Artículos para el hogar
6	7 Deportes	Productos y accesorios relacionados con deportes y actividades al aire libre.

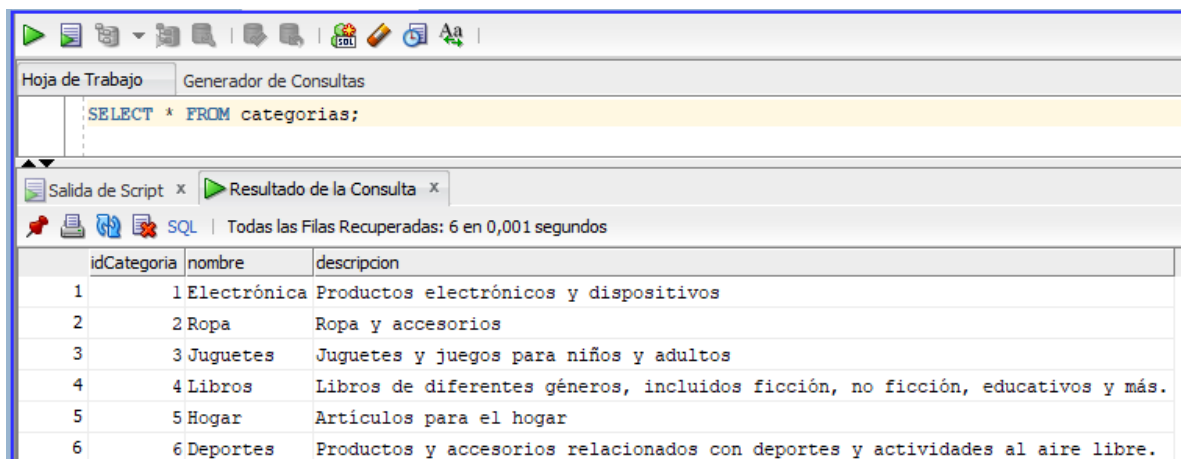
c)

Comprobamos que no se esta utilizando las categorías Hogar y Deportes, en otras tablas.

```
SELECT * FROM productos WHERE idCategoria = 6 OR idCategoria = 7;
```

Al comprobar que no se usa, realizamos los cambios oportunos.

```
UPDATE categorias SET idCategoria = 5 WHERE idCategoria = 6;
UPDATE categorias SET idCategoria = 6 WHERE idCategoria = 7;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT * FROM categorias;
```

Salida de Script x | Resultado de la Consulta x

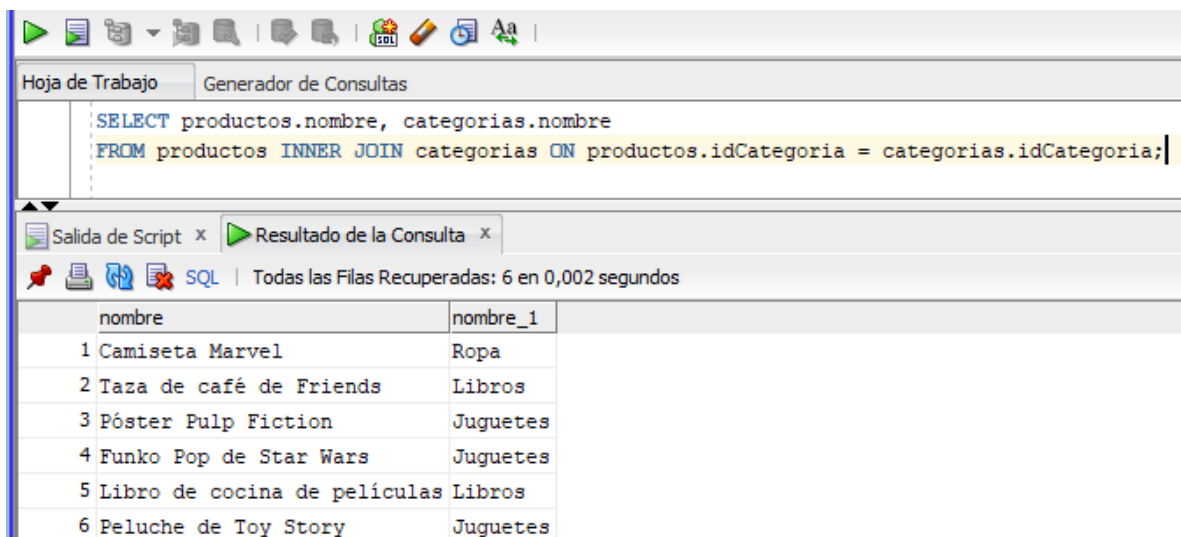
Todas las Filas Recuperadas: 6 en 0,001 segundos

idCategoria	nombre	descripcion
1	1 Electrónica	Productos electrónicos y dispositivos
2	2 Ropa	Ropa y accesorios
3	3 Juguetes	Juguetes y juegos para niños y adultos
4	4 Libros	Libros de diferentes géneros, incluidos ficción, no ficción, educativos y más.
5	5 Hogar	Artículos para el hogar
6	6 Deportes	Productos y accesorios relacionados con deportes y actividades al aire libre.

d)

Comprobamos como aparecen los productos y sus categorías.

```
SELECT productos.nombre, categorias.nombre
FROM productos INNER JOIN categorias ON productos.idCategoria =
categorias.idCategoria;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT productos.nombre, categorias.nombre
FROM productos INNER JOIN categorias ON productos.idCategoria = categorias.idCategoria;
```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 6 en 0,002 segundos

	nombre	nombre_1
1	Camiseta Marvel	Ropa
2	Taza de café de Friends	Libros
3	Póster Pulp Fiction	Juguetes
4	Funko Pop de Star Wars	Juguetes
5	Libro de cocina de películas	Libros
6	Peluche de Toy Story	Juguetes

Se procede a cambiar el producto "Taza de café de Friends" a la categoría "Hogar"

```
UPDATE productos SET idCategoria = 5 WHERE idProducto = 2;
```

Volvemos a comprobar que ya están correctas las categorías en los productos.

Hoja de Trabajo Generador de Consultas

```
SELECT productos.idProducto, productos.nombre, categorias.nombre
FROM productos INNER JOIN categorias ON productos.idCategoria = categorias.idCategoria;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 6 en 0,001 segundos

	idProducto	nombre	nombre_1
1	1	Camiseta Marvel	Ropa
2	2	Taza de café de Friends	Hogar
3	3	Póster Pulp Fiction	Juguetes
4	4	Funko Pop de Star Wars	Juguetes
5	5	Libro de cocina de películas	Libros
6	6	Peluche de Toy Story	Juguetes

Ejercicio 2

1.- Obtén una lista de todos los clientes.

```
SELECT * FROM CLIENTES;
```

MySQL Xampp

Hoja de Trabajo Generador de Consultas

```
SELECT * FROM CLIENTES;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 5 en 0,001 segundos

	idCliente	nombre	apellido	email	telefono	direccion
1	1	Juan	Pérez	juan.perez@email.com	+1234567890	Calle 123, Ciudad A
2	2	Maria	González	maria.gonzalez@email.com	+2345678901	Avenida 456, Ciudad B
3	3	Carlos	Ramírez	carlos.ramirez@email.com	+3456789012	Calle 789, Ciudad C
4	4	Ana	Rodríguez	ana.rodriguez@email.com	+4567890123	Avenida 321, Ciudad D
5	5	Pedro	López	pedro.lopez@email.com	+5678901234	Calle 654, Ciudad E

2.- Encuentra todos los productos con un precio mayor a 30.

```
SELECT * FROM PRODUCTOS WHERE precio > 30;
```

MySQL Xampp

Hoja de Trabajo Generador de Consultas

```
SELECT * FROM PRODUCTOS WHERE precio > 30;
```

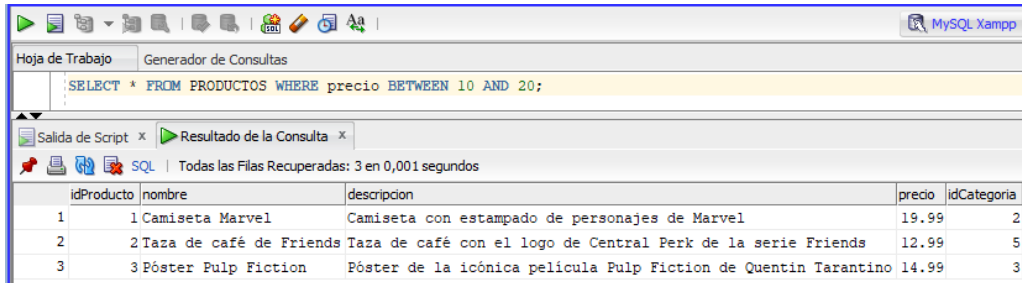
Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,001 segundos

	idProducto	nombre	descripcion	precio	idCategoria
1	6	Peluche de Toy Story	Peluche de un personaje de la película animada Toy Story	34.99	3

3.- Encuentra todos los productos con un precio entre 10 a 20.

```
SELECT * FROM PRODUCTOS WHERE precio BETWEEN 10 AND 20;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT * FROM PRODUCTOS WHERE precio BETWEEN 10 AND 20;
```

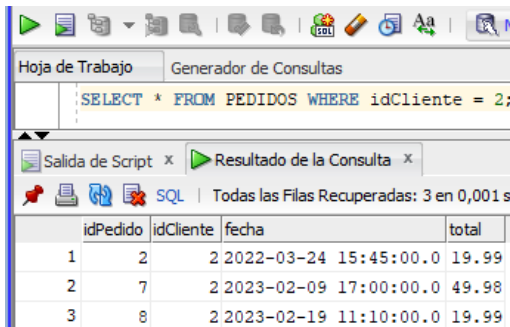
Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,001 segundos

idProducto	nombre	descripcion	precio	idCategoria
1	1 Camiseta Marvel	Camiseta con estampado de personajes de Marvel	19.99	2
2	2 Taza de café de Friends	Taza de café con el logo de Central Perk de la serie Friends	12.99	5
3	3 Póster Pulp Fiction	Póster de la icónica película Pulp Fiction de Quentin Tarantino	14.99	3

4.- Muestra todos los pedidos realizados por el cliente con el id 2.

```
SELECT * FROM PEDIDOS WHERE idCliente = 2;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT * FROM PEDIDOS WHERE idCliente = 2;
```

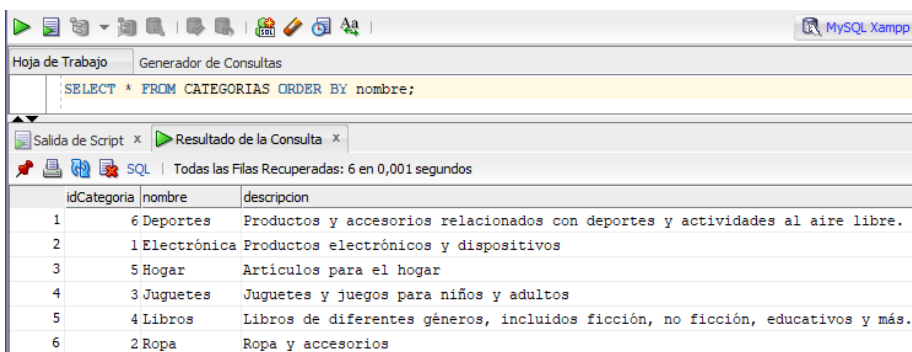
Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,001 s

idPedido	idCliente	fecha	total
1	2	2 2022-03-24 15:45:00.0	19.99
2	7	2 2023-02-09 17:00:00.0	49.98
3	8	2 2023-02-19 11:10:00.0	19.99

5.- Obtén una lista de todas las categorías ordenadas alfabéticamente.

```
SELECT * FROM CATEGORIAS ORDER BY nombre;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT * FROM CATEGORIAS ORDER BY nombre;
```

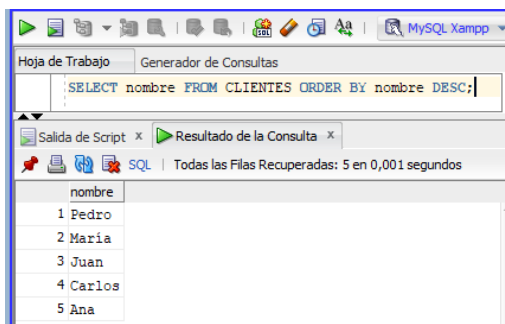
Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 6 en 0,001 segundos

idCategoria	nombre	descripcion
1	6 Deportes	Productos y accesorios relacionados con deportes y actividades al aire libre.
2	1 Electrónica	Productos electrónicos y dispositivos
3	5 Hogar	Articulos para el hogar
4	3 Juguetes	Juguetes y juegos para niños y adultos
5	4 Libros	Libros de diferentes géneros, incluidos ficción, no ficción, educativos y más.
6	2 Ropa	Ropa y accesorios

6.- Obtén los nombres de los clientes ordenado alfabéticamente de forma descendente [Z-A] .

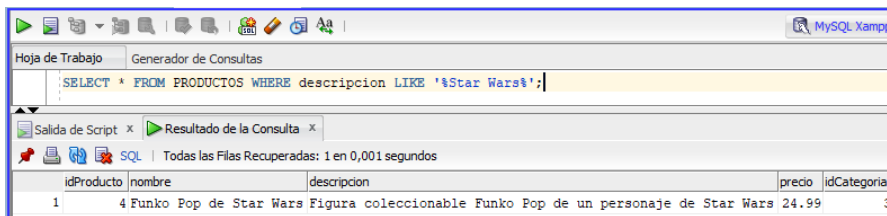
```
SELECT nombre FROM CLIENTES ORDER BY nombre DESC;
```



nombre
1 Pedro
2 Maria
3 Juan
4 Carlos
5 Ana

7.- Encuentra todos los productos cuya descripción contenga la palabra 'Star Wars'.

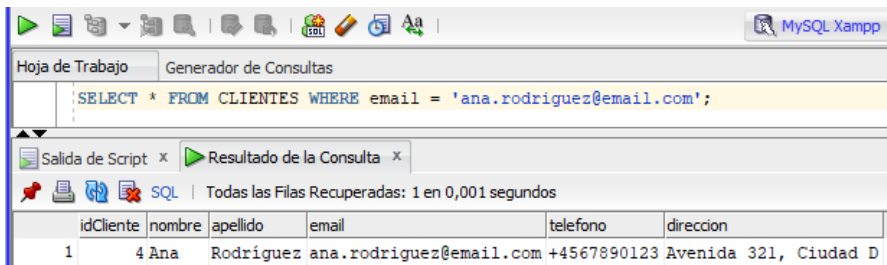
```
SELECT * FROM PRODUCTOS WHERE descripcion LIKE '%Star Wars%';
```



idProducto	nombre	descripcion	precio	idCategoria
1	4 Funko Pop de Star Wars	Figura coleccionable Funko Pop de un personaje de Star Wars	24.99	3

8.- Encuentra el cliente con el email 'ana.rodriguez@email.com'.

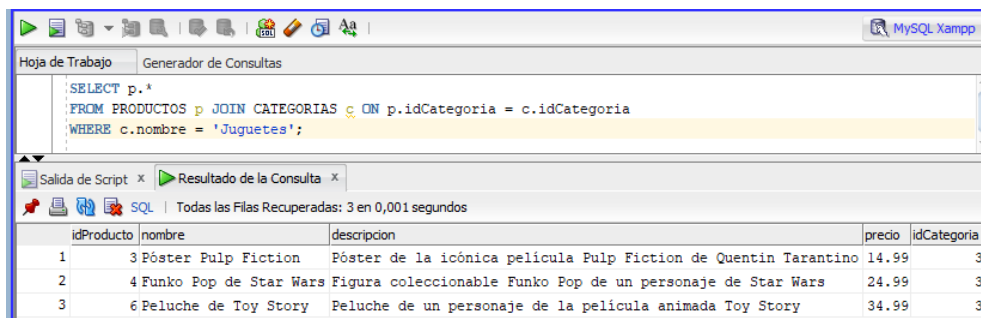
```
SELECT * FROM CLIENTES WHERE email = 'ana.rodriguez@email.com';
```



idCliente	nombre	apellido	email	telefono	direccion
1	4 Ana	Rodriguez	ana.rodriguez@email.com	+4567890123	Avenida 321, Ciudad D

9.- Muestra todos los productos que pertenecen a la categoría con el nombre 'Juguetes'.

```
SELECT p.*
FROM PRODUCTOS p
JOIN CATEGORIAS c ON p.idCategoria = c.idCategoria
WHERE c.nombre = 'Juguetes';
```



Hoja de Trabajo Generador de Consultas

```
SELECT p.*
FROM PRODUCTOS p JOIN CATEGORIAS c ON p.idCategoria = c.idCategoria
WHERE c.nombre = 'Juguetes';
```

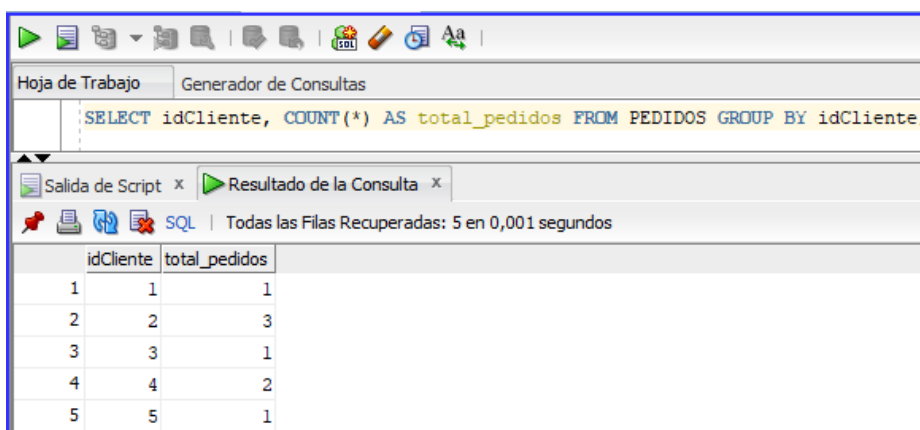
Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,001 segundos

idProducto	nombre	descripcion	precio	idCategoria
1	3 Póster Pulp Fiction	Póster de la icónica película Pulp Fiction de Quentin Tarantino	14.99	3
2	4 Funko Pop de Star Wars	Figura coleccionable Funko Pop de un personaje de Star Wars	24.99	3
3	6 Peluche de Toy Story	Peluche de un personaje de la película animada Toy Story	34.99	3

10.- Obtén el total de pedidos por cada cliente.

```
SELECT idCliente, COUNT(*) AS total_pedidos FROM PEDIDOS GROUP BY idCliente;
```



Hoja de Trabajo Generador de Consultas

```
SELECT idCliente, COUNT(*) AS total_pedidos FROM PEDIDOS GROUP BY idCliente;
```

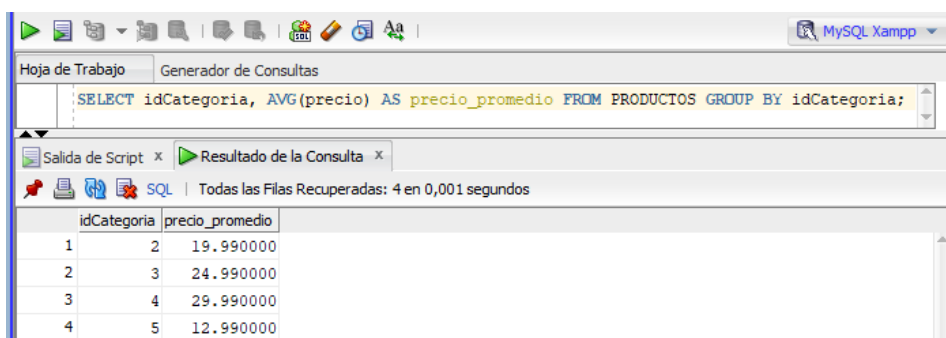
Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 5 en 0,001 segundos

idCliente	total_pedidos
1	1
2	3
3	1
4	2
5	1

11.- Encuentra el precio promedio de los productos por categoría.

```
SELECT idCategoria, AVG(precio) AS precio_promedio FROM PRODUCTOS GROUP BY idCategoria;
```



Hoja de Trabajo Generador de Consultas

```
SELECT idCategoria, AVG(precio) AS precio_promedio FROM PRODUCTOS GROUP BY idCategoria;
```

Salida de Script x Resultado de la Consulta x

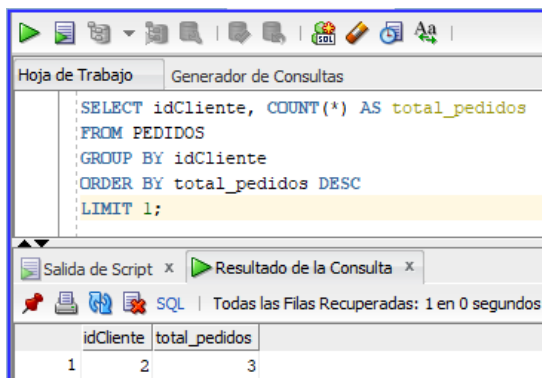
Todas las Filas Recuperadas: 4 en 0,001 segundos

idCategoria	precio_promedio
1	2 19.990000
2	3 24.990000
3	4 29.990000
4	5 12.990000

12.- Encuentra el cliente que ha realizado más pedidos.

```
SELECT idCliente, COUNT(*) AS total_pedidos
FROM PEDIDOS
GROUP BY idCliente
ORDER BY total_pedidos DESC
```


LIMIT 1;



Hoja de Trabajo | Generador de Consultas

```

SELECT idCliente, COUNT(*) AS total_pedidos
FROM PEDIDOS
GROUP BY idCliente
ORDER BY total_pedidos DESC
LIMIT 1;

```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0 segundos

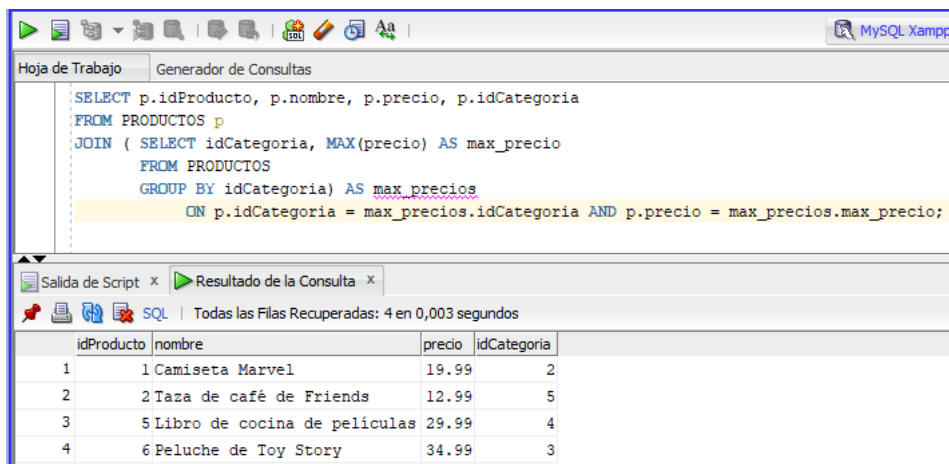
idCliente	total_pedidos
1	3

13.- Encuentra el producto más caro en cada categoría.

```

SELECT p.idProducto, p.nombre, p.precio, p.idCategoria
FROM PRODUCTOS p
JOIN (
    SELECT idCategoria, MAX(precio) AS max_precio
    FROM PRODUCTOS
    GROUP BY idCategoria
) AS max_precios ON p.idCategoria = max_precios.idCategoria AND p.precio =
max_precios.max_precio;

```



Hoja de Trabajo | Generador de Consultas

```

SELECT p.idProducto, p.nombre, p.precio, p.idCategoria
FROM PRODUCTOS p
JOIN ( SELECT idCategoria, MAX(precio) AS max_precio
      FROM PRODUCTOS
      GROUP BY idCategoria) AS max_precios
ON p.idCategoria = max_precios.idCategoria AND p.precio = max_precios.max_precio;

```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 4 en 0,003 segundos

idProducto	nombre	precio	idCategoria
1	1 Camiseta Marvel	19.99	2
2	2 Taza de café de Friends	12.99	5
3	5 Libro de cocina de películas	29.99	4
4	6 Peluche de Toy Story	34.99	3

14.- Obtén el total de ventas por mes y año.

```

SELECT YEAR(fecha) AS year, MONTH(fecha) AS month, SUM(total) AS ventas_totales
FROM PEDIDOS
GROUP BY year, month
ORDER BY year, month;

```

The screenshot shows the MySQL Xampp interface. The SQL editor contains the following query:

```
SELECT YEAR(fecha) AS year, MONTH(fecha) AS month, SUM(total) AS ventas_totales
FROM PEDIDOS
GROUP BY year, month
ORDER BY year, month;
```

The results pane shows the following data:

	year	month	ventas_totales
1	2022	1	549.98
2	2022	3	19.99
3	2022	4	319.98
4	2022	5	99.98
5	2022	12	49.99
6	2023	1	137.94
7	2023	2	69.97

15.- Muestra el número de productos por categoría y su precio promedio, pero solo para las categorías con 2 o más productos.

```
SELECT idCategoria, COUNT(*) AS num_productos, AVG(precio) AS precio_promedio
FROM PRODUCTOS
GROUP BY idCategoria
HAVING num_productos >= 2;
```

The screenshot shows the MySQL Xampp interface. The SQL editor contains the following query:

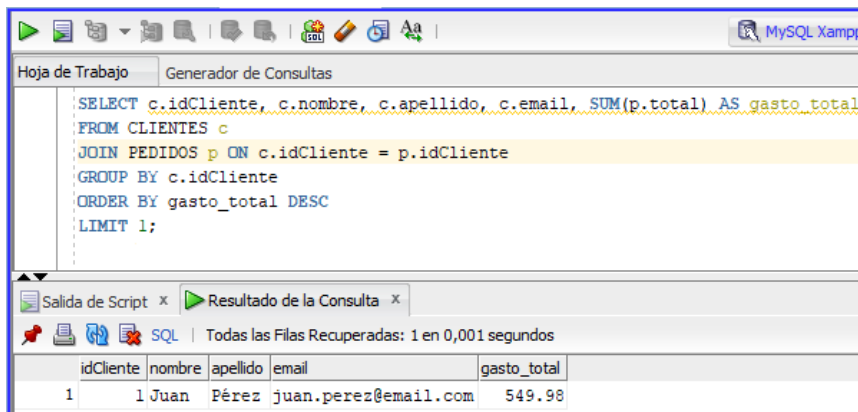
```
SELECT idCategoria, COUNT(*) AS num_productos, AVG(precio) AS precio_promedio
FROM PRODUCTOS
GROUP BY idCategoria
HAVING num_productos >= 2;
```

The results pane shows the following data:

idCategoria	num_productos	precio_promedio
1	3	24.990000

16.- Encuentra el cliente con el mayor gasto total en todos sus pedidos, incluyendo sus datos personales y el total gastado.

```
SELECT c.idCliente, c.nombre, c.apellido, c.email, SUM(p.total) AS gasto_total
FROM CLIENTES c
JOIN PEDIDOS p ON c.idCliente = p.idCliente
GROUP BY c.idCliente
ORDER BY gasto_total DESC
LIMIT 1;
```



MySQL Xampp

Hoja de Trabajo | Generador de Consultas

```

SELECT c.idCliente, c.nombre, c.apellido, c.email, SUM(p.total) AS gasto_total
FROM CLIENTES c
JOIN PEDIDOS p ON c.idCliente = p.idCliente
GROUP BY c.idCliente
ORDER BY gasto_total DESC
LIMIT 1;

```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0,001 segundos

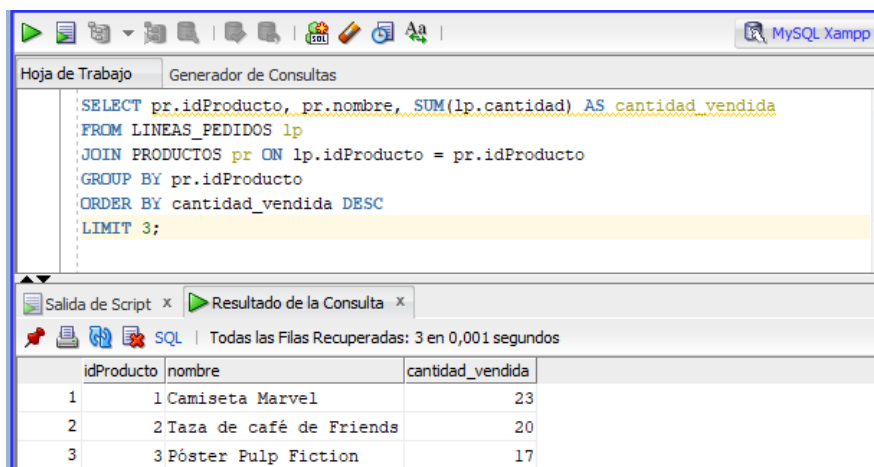
idCliente	nombre	apellido	email	gasto_total
1	Juan	Pérez	juan.perez@email.com	549.98

17.- Encuentra los tres productos más vendidos (asumiendo la cantidad total de cada línea de pedidos).

```

SELECT pr.idProducto, pr.nombre, SUM(lp.cantidad) AS cantidad_vendida
FROM LINEAS_PEDIDOS lp
      JOIN PRODUCTOS pr ON lp.idProducto = pr.idProducto
GROUP BY pr.idProducto
ORDER BY cantidad_vendida DESC
LIMIT 3;

```



MySQL Xampp

Hoja de Trabajo | Generador de Consultas

```

SELECT pr.idProducto, pr.nombre, SUM(lp.cantidad) AS cantidad_vendida
FROM LINEAS_PEDIDOS lp
JOIN PRODUCTOS pr ON lp.idProducto = pr.idProducto
GROUP BY pr.idProducto
ORDER BY cantidad_vendida DESC
LIMIT 3;

```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,001 segundos

idProducto	nombre	cantidad_vendida
1	1 Camiseta Marvel	23
2	2 Taza de café de Friends	20
3	3 Póster Pulp Fiction	17

18.- Calcula el total gastado en cada categoría de productos por todos los clientes.

```

SELECT c.idCategoria, c.nombre, SUM(lp.total) AS gasto_total
FROM PEDIDOS pe
      JOIN LINEAS_PEDIDOS lp ON pe.idPedido = lp.idPedido
      JOIN PRODUCTOS p ON lp.idProducto = p.idProducto
      JOIN CATEGORIAS c ON p.idCategoria = c.idCategoria
GROUP BY c.idCategoria;

```

MySQL Xampp

Hoja de Trabajo Generador de Consultas

```

SELECT c.idCategoria, c.nombre, SUM(lp.total) AS gasto_total
FROM PEDIDOS p
JOIN LINEAS_PEDIDOS lp ON pe.idPedido = lp.idPedido
JOIN PRODUCTOS p ON lp.idProducto = p.idProducto
JOIN CATEGORIAS c ON p.idCategoria = c.idCategoria
GROUP BY c.idCategoria;

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 4 en 0,001 segundos

idCategoria	nombre	gasto_total
1	2 Ropa	415.11
2	3 Juguetes	516.94
3	4 Libros	69.97
4	5 Hogar	572.96

19.- Muestra los clientes que han comprado productos de al menos tres categorías diferentes.

```

SELECT cl.idCliente, cl.nombre, cl.apellido, COUNT(DISTINCT p.idCategoria) AS num_categorias
FROM CLIENTES cl
JOIN PEDIDOS pe ON cl.idCliente = pe.idCliente
JOIN LINEAS_PEDIDOS lp ON pe.idPedido = lp.idPedido
JOIN PRODUCTOS p ON lp.idProducto = p.idProducto
GROUP BY cl.idCliente
HAVING num_categorias >= 3;

```

MySQL Xampp

Hoja de Trabajo Generador de Consultas

```

SELECT cl.idCliente, cl.nombre, cl.apellido, COUNT(DISTINCT p.idCategoria) AS num_categorias
FROM CLIENTES cl
JOIN PEDIDOS pe ON cl.idCliente = pe.idCliente
JOIN LINEAS_PEDIDOS lp ON pe.idPedido = lp.idPedido
JOIN PRODUCTOS p ON lp.idProducto = p.idProducto
GROUP BY cl.idCliente
HAVING num_categorias >= 3;

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 2 en 0,002 segundos

idCliente	nombre	apellido	num_categorias
1	2 Maria	González	3
2	5 Pedro	López	4

20.- Encuentra el mes con las mayores ventas totales. Y el mes con las menores ventas totales. Tiene que aparecer cada información en una fila diferente.

```
(SELECT YEAR(pe.fecha) AS year, MONTH(pe.fecha) AS month, SUM(pe.total) AS
ventas_totales
FROM PEDIDOS pe
GROUP BY year, month
ORDER BY ventas_totales DESC
LIMIT 1)
UNION
(SELECT YEAR(pe.fecha) AS year, MONTH(pe.fecha) AS month, SUM(pe.total) AS
ventas_totales
FROM PEDIDOS pe
GROUP BY year, month
ORDER BY ventas_totales
LIMIT 1);
```

The screenshot shows the MySQL Xampp interface. The top part displays the SQL query entered in the 'Generador de Consultas' (Query Generator) tab. The bottom part shows the 'Resultado de la Consulta' (Query Result) tab, which displays the results of the query in a table format. The table has three columns: 'year', 'month', and 'ventas_totales'. There are two rows of data, representing the months with the highest and lowest total sales.

year	month	ventas_totales
1 2022	1	549.98
2 2022	3	19.99

Unidad 6

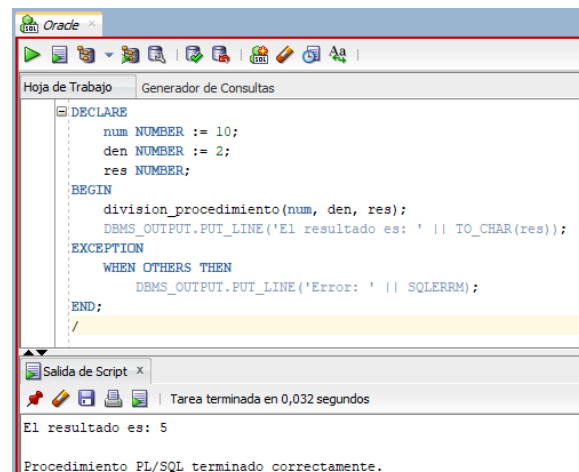
Ejercicio 1 de la videoclase

Crea un procedimiento o función que realice una división.

PROCEDIMIENTO

```
CREATE OR REPLACE PROCEDURE division_procedimiento(numerador IN NUMBER,
denominador IN NUMBER, resultado OUT NUMBER) IS
BEGIN
    IF denominador = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Denominador no puede ser cero');
    ELSE
        resultado := numerador / denominador;
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END division_procedimiento;
/

-- EJECUTAR EL PROCEDIMIENTO
DECLARE
    num NUMBER := 10;
    den NUMBER := 2;
    res NUMBER;
BEGIN
    division_procedimiento(num, den, res);
    DBMS_OUTPUT.PUT_LINE('El resultado es: ' || TO_CHAR(res));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

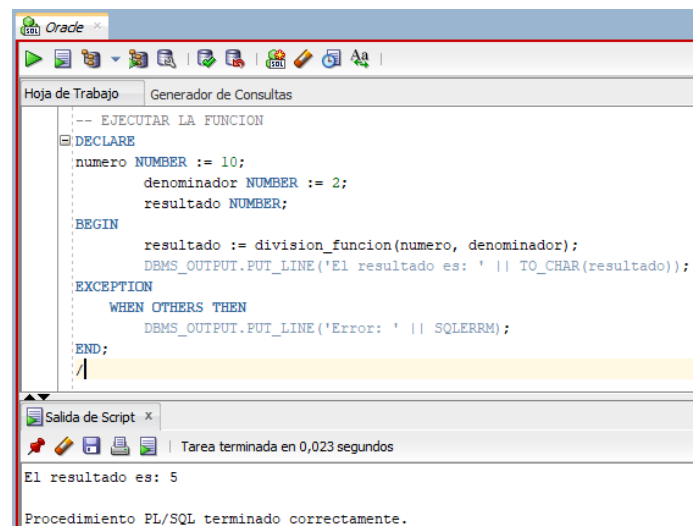




FUNCIÓN

```
CREATE OR REPLACE FUNCTION division_funcion(numerador IN NUMBER, denominador IN
NUMBER) RETURN NUMBER IS
    resultado NUMBER(10, 2);
BEGIN
    IF denominador = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Denominador no puede ser cero');
    ELSE
        resultado := numerador / denominador;
    END IF;
    RETURN resultado;
END division_funcion;
/

-- EJECUTAR LA FUNCION
DECLARE
    numero NUMBER := 10;
    denominador NUMBER := 2;
    resultado NUMBER;
BEGIN
    resultado := division_funcion(numero, denominador);
    DBMS_OUTPUT.PUT_LINE('El resultado es: ' || TO_CHAR(resultado));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



Ejercicio 2 de la videoclase

Crea un procedimiento o función que pida un nombre y edad. Muestre el nombre y si es mayor de edad o no.

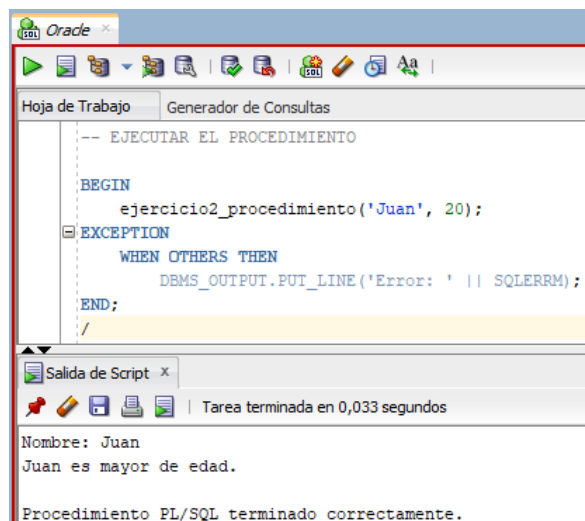
PROCEDIMIENTO

```
CREATE OR REPLACE PROCEDURE ejercicio2_procedimiento (nombre IN VARCHAR2, edad
IN NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Nombre: ' || nombre);
    IF edad >= 18 THEN
        DBMS_OUTPUT.PUT_LINE(nombre || ' es mayor de edad.');
```

```
    ELSE
        DBMS_OUTPUT.PUT_LINE(nombre || ' no es mayor de edad.');
```

```
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ejercicio2_procedimiento;
/

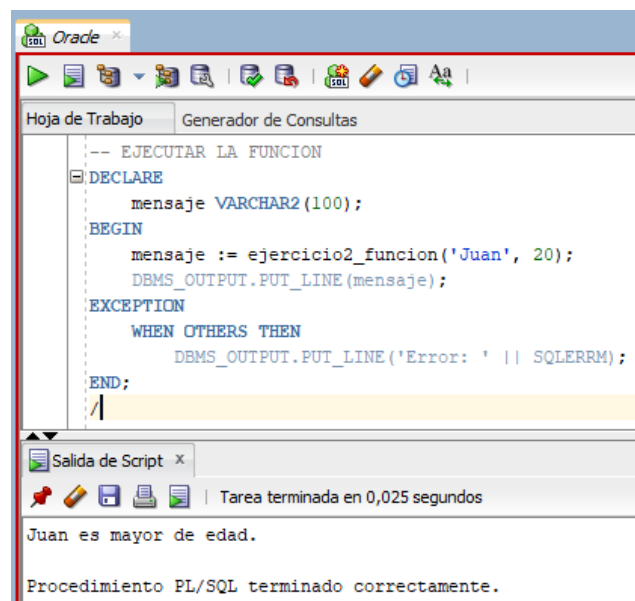
-- EJECUTAR EL PROCEDIMIENTO
BEGIN
    ejercicio2_procedimiento('Juan', 20);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



FUNCIÓN

```
CREATE OR REPLACE FUNCTION ejercicio2_funcion(nombre IN VARCHAR2, edad IN
NUMBER) RETURN VARCHAR2 IS
    resultado VARCHAR2(100);
BEGIN
    IF edad >= 18 THEN
        resultado := nombre || ' es mayor de edad.';
    ELSE
        resultado := nombre || ' no es mayor de edad.';
    END IF;
    RETURN resultado;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        RETURN NULL;
END ejercicio2_funcion;
/

-- EJECUTAR LA FUNCION
DECLARE
    mensaje VARCHAR2(100);
BEGIN
    mensaje := ejercicio2_funcion('Juan', 20);
    DBMS_OUTPUT.PUT_LINE(mensaje);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



The screenshot shows the Oracle SQL Developer application. The main window is titled "Orade" and has a toolbar with various icons. Below the toolbar, there are two tabs: "Hoja de Trabajo" (Worksheet) and "Generador de Consultas" (Query Generator). The "Hoja de Trabajo" tab is active, displaying the PL/SQL code from the previous block. The code is highlighted in blue and green. At the bottom of the window, there is a "Salida de Script" (Script Output) window. It shows the output of the execution: "Juan es mayor de edad." and "Procedimiento PL/SQL terminado correctamente." (PL/SQL procedure terminated successfully). The output window also indicates that the task was completed in 0.025 seconds.



Ejercicio 3 de la videoclase

Crea un procedimiento o función que muestre todos los campos de una tabla. (Podéis crear la tabla que queráis)

```
-- TABLA
CREATE TABLE Empleados (
    idEmpleado NUMBER(6),
    nombre VARCHAR2(20),
    apellido VARCHAR2(25),
    fechaContratacion DATE,
    salario NUMBER(8, 2)
);

-- DATOS
INSERT INTO Empleados (idEmpleado, nombre, apellido, fechaContratacion, salario)
VALUES (1, 'Juan', 'Pérez', TO_DATE('2020-03-01','YYYY-MM-DD'), 5000.00);

INSERT INTO Empleados (idEmpleado, nombre, apellido, fechaContratacion, salario)
VALUES (2, 'Ana', 'González', TO_DATE('2018-07-15','YYYY-MM-DD'), 5500.00);

INSERT INTO Empleados (idEmpleado, nombre, apellido, fechaContratacion, salario)
VALUES (3, 'Pedro', 'García', TO_DATE('2019-10-30','YYYY-MM-DD'), 6000.00);

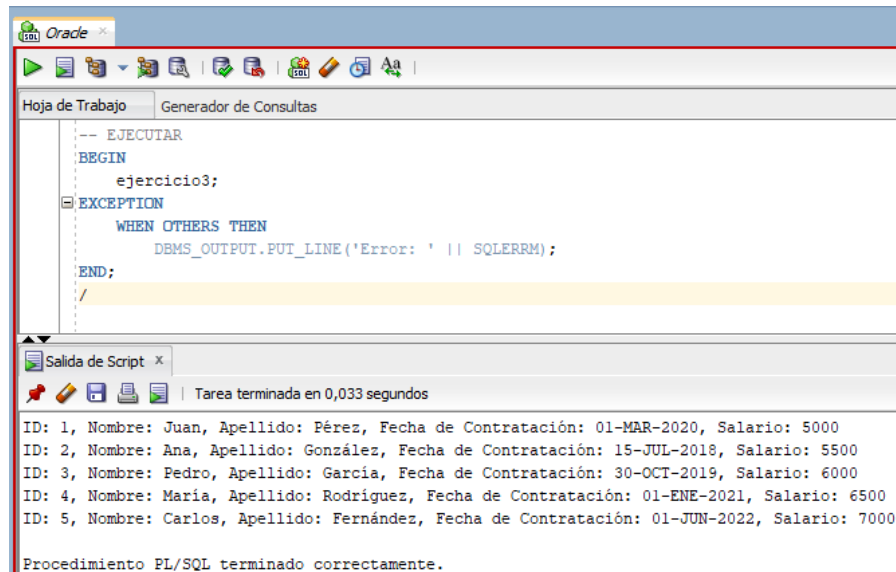
INSERT INTO Empleados (idEmpleado, nombre, apellido, fechaContratacion, salario)
VALUES (4, 'María', 'Rodríguez', TO_DATE('2021-01-01','YYYY-MM-DD'), 6500.00);

INSERT INTO Empleados (idEmpleado, nombre, apellido, fechaContratacion, salario)
VALUES (5, 'Carlos', 'Fernández', TO_DATE('2022-06-01','YYYY-MM-DD'), 7000.00);

CREATE OR REPLACE PROCEDURE ejercicio3 IS
    cursor c_empleados is SELECT * FROM Empleados;
    registro_empleado c_empleados%ROWTYPE;
BEGIN
    OPEN c_empleados;
    LOOP
        FETCH c_empleados INTO registro_empleado;
        EXIT WHEN c_empleados%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('ID: ' || registro_empleado.idEmpleado || ',
Nombre: ' || registro_empleado.nombre || ', Apellido: ' ||
registro_empleado.apellido || ', Fecha de Contratación: ' ||
TO_CHAR(registro_empleado.fechaContratacion, 'DD-MON-YYYY') || ', Salario: ' ||
registro_empleado.salario);
    END LOOP;
    CLOSE c_empleados;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END ejercicio3;
/
```



```
-- EJECUTAR
BEGIN
    ejercicio3;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



Ejercicio 4 de la videoclase

Crea una tabla *LINEA_FACTURA* (*numero*, *nombre_producto*, *cantidad*, *precio_unitario* y *precio_total*)

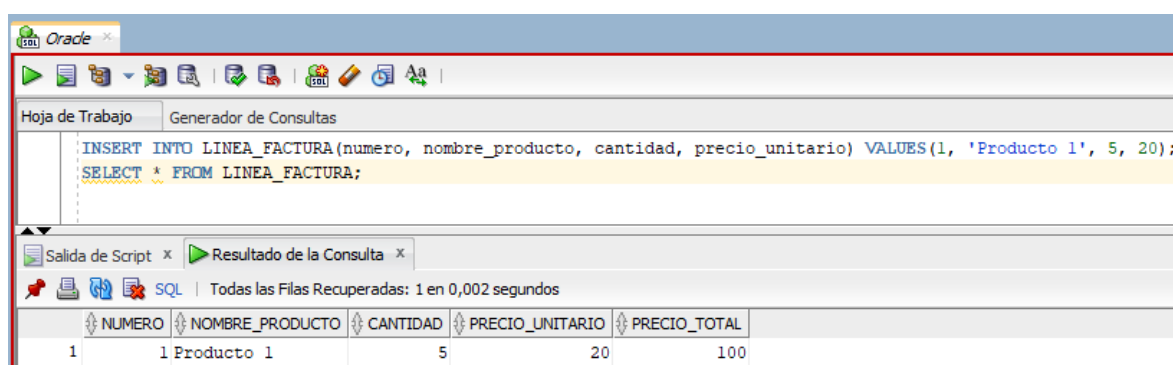
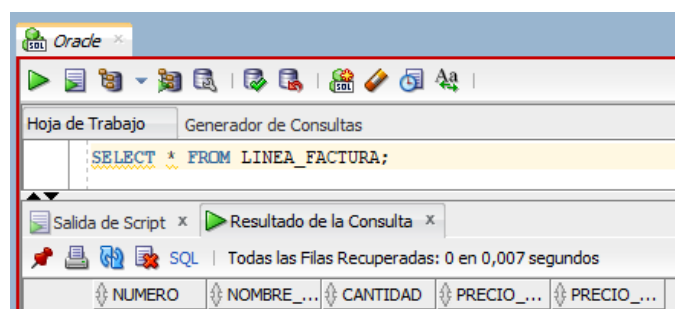
Se pide un trigger que cuando se inserte un valor, calcule el precio total. El precio total es cantidad * precio_unitario

```
-- Tabla
CREATE TABLE LINEA_FACTURA (
    numero NUMBER,
    nombre_producto VARCHAR2(100),
    cantidad NUMBER,
    precio_unitario NUMBER,
    precio_total NUMBER
);

-- Trigger
CREATE OR REPLACE TRIGGER calcular_precio_total
BEFORE INSERT ON LINEA_FACTURA
FOR EACH ROW
BEGIN
    :NEW.precio_total := :NEW.cantidad * :NEW.precio_unitario;
END;
/
```

-- Prueba

```
SELECT * FROM LINEA_FACTURA;  
INSERT INTO LINEA_FACTURA(numero, nombre_producto, cantidad, precio_unitario)  
VALUES(1, 'Producto 1', 5, 20);  
SELECT * FROM LINEA_FACTURA;
```





Unidad 7

Ejercicio 1 de la videoclase

Realiza una copia de la base de datos. Elimina la base de datos. Vuelve a restaurarla.

Utilizaremos la siguiente base de datos.

```
CREATE DATABASE prueba;
```

```
USE prueba;
```

```
CREATE TABLE Usuarios (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    password VARCHAR(50) NOT NULL  
);
```

```
INSERT INTO Usuarios (id, nombre, email, password) VALUES  
(1, 'Juan', 'juan@example.com', 'contraseña123'),  
(2, 'María', 'maria@example.com', 'contraseña456'),  
(3, 'Pedro', 'pedro@example.com', 'contraseña789');
```

```
CREATE TABLE Productos (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    descripcion TEXT,  
    precio DECIMAL(8, 2) NOT NULL  
);
```

```
INSERT INTO Productos (id, nombre, descripcion, precio) VALUES  
(1, 'Camiseta', 'Camiseta de algodón con estampado', 19.99),  
(2, 'Pantalón', 'Pantalón vaquero ajustado', 39.99),  
(3, 'Zapatos', 'Zapatos de cuero con suela de goma', 59.99);
```

```
CREATE TABLE Pedidos (  
    id INT PRIMARY KEY,  
    id_usuario INT NOT NULL,  
    fecha TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id)  
);
```

```
INSERT INTO Pedidos (id, id_usuario) VALUES  
(1, 1),  
(2, 2),  
(3, 3);
```



```
CREATE TABLE DetallesPedido (  
    id INT PRIMARY KEY,  
    id_pedido INT NOT NULL,  
    id_producto INT NOT NULL,  
    cantidad INT NOT NULL,  
    precio_unitario DECIMAL(8, 2) NOT NULL,  
    FOREIGN KEY (id_pedido) REFERENCES Pedidos(id),  
    FOREIGN KEY (id_producto) REFERENCES Productos(id)  
);  
  
INSERT INTO DetallesPedido (id, id_pedido, id_producto, cantidad,  
precio_unitario) VALUES  
    (1, 1, 1, 2, 19.99),  
    (2, 1, 3, 1, 59.99),  
    (3, 2, 2, 1, 39.99),  
    (4, 3, 1, 3, 19.99),  
    (5, 3, 3, 2, 59.99);  
  
CREATE TABLE Comentarios (  
    id INT PRIMARY KEY,  
    id_usuario INT NOT NULL,  
    id_producto INT NOT NULL,  
    comentario TEXT NOT NULL,  
    fecha TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id),  
    FOREIGN KEY (id_producto) REFERENCES Productos(id)  
);  
  
INSERT INTO Comentarios (id, id_usuario, id_producto, comentario) VALUES  
    (1, 1, 1, 'Muy buena calidad de la camiseta'),  
    (2, 2, 2, 'Los pantalones son muy cómodos'),  
    (3, 3, 3, 'Los zapatos me han durado mucho tiempo');
```

Realizamos la copia mediante el terminal con la herramienta mysqldump.

```
mysqldump --port=3360 --user=user --password=user prueba >  
C:\COPIAS\copia20230105.prueba.sql
```

```
C:\mysql\bin>mysqldump --port=3360 --user=user --password=user prueba>C:\COPIAS\copia20230105.prueba.sql  
C:\mysql\bin>
```

En la instrucción indicamos el puerto que usa el servidor de la base de datos, así como el usuario y contraseña. La última parte se especifica que base de datos se copiará, en este caso, "prueba" y donde se ubicará la copia .sql, en C:\COPIAS\

Revisamos el fichero generado.

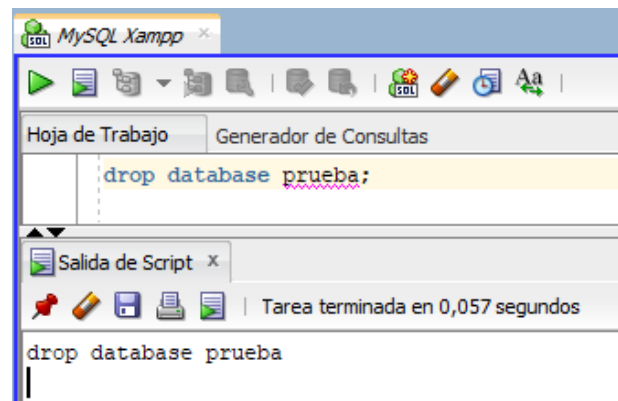
> Disco local (C:) > COPIAS

Nombre	Fecha de modificación	Tipo	Tamaño
copia20230105.prueba.sql	05/05/2023 11:12	Archivo SQL	7 KB


```

C:\COPIAS\copia20230105.prueba.sql - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Complementos  Pestañas  ?  +
1  -- MariaDB dump 10.19  Distrib 10.4.24-MariaDB, for Win64 (AMD64)
2
3  -- Host: localhost    Database: prueba
4  -- Server version  10.4.24-MariaDB
5
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10 /*!40101 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17
18 --
19 -- Table structure for table `comentarios`
20 --
21
22 DROP TABLE IF EXISTS `comentarios`;
23 /*!40101 SET @saved_cs_client = @@character_set_client */;
24 /*!40101 SET character_set_client = utf8 */;
25 CREATE TABLE `comentarios` (
26   `id` int(11) NOT NULL,
27   `id_usuario` int(11) NOT NULL,
28   `id_producto` int(11) NOT NULL,
29   `comentario` text NOT NULL,
30   `fecha` timestamp NOT NULL DEFAULT current_timestamp(),
31   PRIMARY KEY (`id`),
32   KEY `id_usuario` (`id_usuario`),
33   KEY `id_producto` (`id_producto`),
34   CONSTRAINT `comentarios_ibfk_1` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id`),
35   CONSTRAINT `comentarios_ibfk_2` FOREIGN KEY (`id_producto`) REFERENCES `productos` (`id`)
26 Structured Query length: 6328  lines: 170  Ln: 1  Col: 1  Pos: 1  Windows (CR LF)  UTF-8  INS
  
```

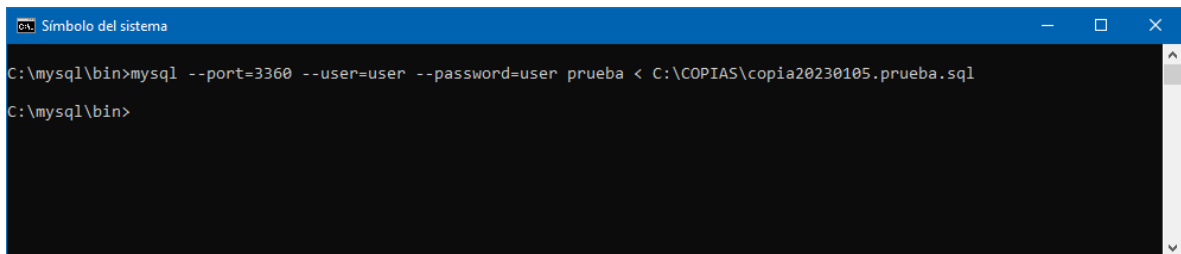
Procedemos a borrar la base de datos, con el comando *DROP DATABASE prueba;*



Ahora recuperamos la base de datos, siguiendo los siguientes pasos:

1. Creamos la base de datos, con el comando, *CREATE DATABASE prueba;*
2. Ejecutamos el comando

```
mysql --port=3360 --user=user --password=user prueba < C:\COPIAS\copia20230105.prueba.sql
```



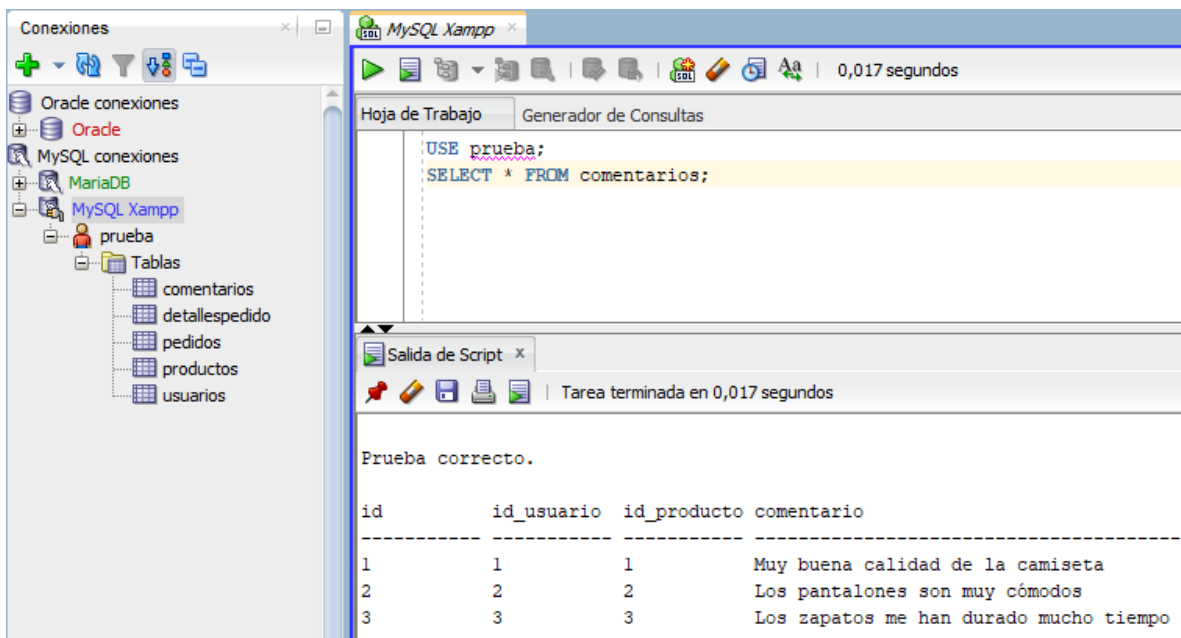
```

C:\mysql\bin>mysql --port=3360 --user=user --password=user prueba < C:\COPIAS\copia20230105.prueba.sql
C:\mysql\bin>

```

En la instrucción indicamos el puerto que usa el servidor de la base de datos, así como el usuario y contraseña. La última parte se especifica la base de datos destino donde se copiará, en este caso, "prueba" y donde está el documento .sql con la copia, en C:\COPIAS\

Comprobamos si se ha restaurado la base de datos y aparecen los datos de las tablas.



MySQL Xampp

Hoja de Trabajo | Generador de Consultas

```
USE prueba;
SELECT * FROM comentarios;
```

Salida de Script x

Tarea terminada en 0,017 segundos

Prueba correcto.

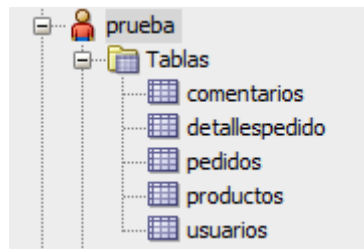
id	id_usuario	id_producto	comentario
1	1	1	Muy buena calidad de la camiseta
2	2	2	Los pantalones son muy cómodos
3	3	3	Los zapatos me han durado mucho tiempo

Ejercicio 2 de la videoclase

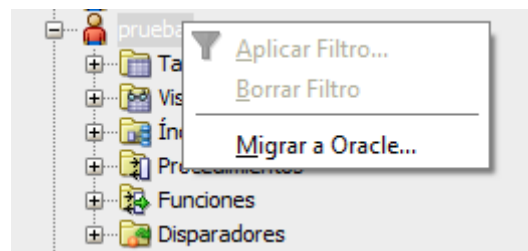
Utiliza el proceso de migración que ofrece SQLDeveloper para traspasar la base de datos MySQL o MariaDB a Oracle

Para realizar este ejercicio, es necesario disponer de dos servidores de base de datos, MySQL y Oracle. Vamos a partir por la misma estructura de la base de datos del ejercicio 1, con las tablas: comentarios, detallespedido, pedidos, productos y usuarios.

La vista que tenemos desde MySQL es la siguiente:

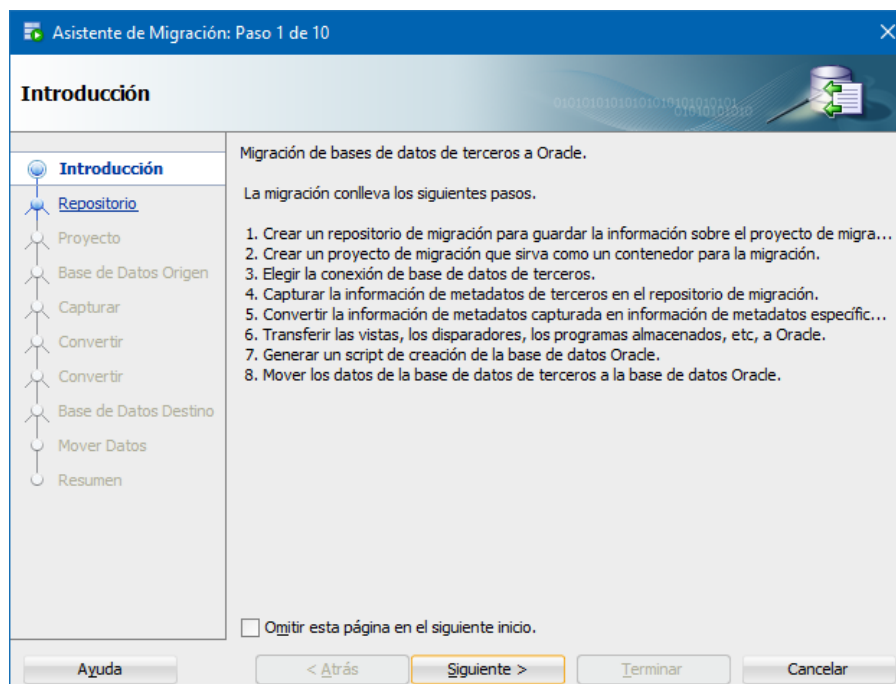


Iniciamos el procedimiento de migración, pulsando botón secundario sobre el nombre de la base de datos y elegir la opción Migrar a Oracle.

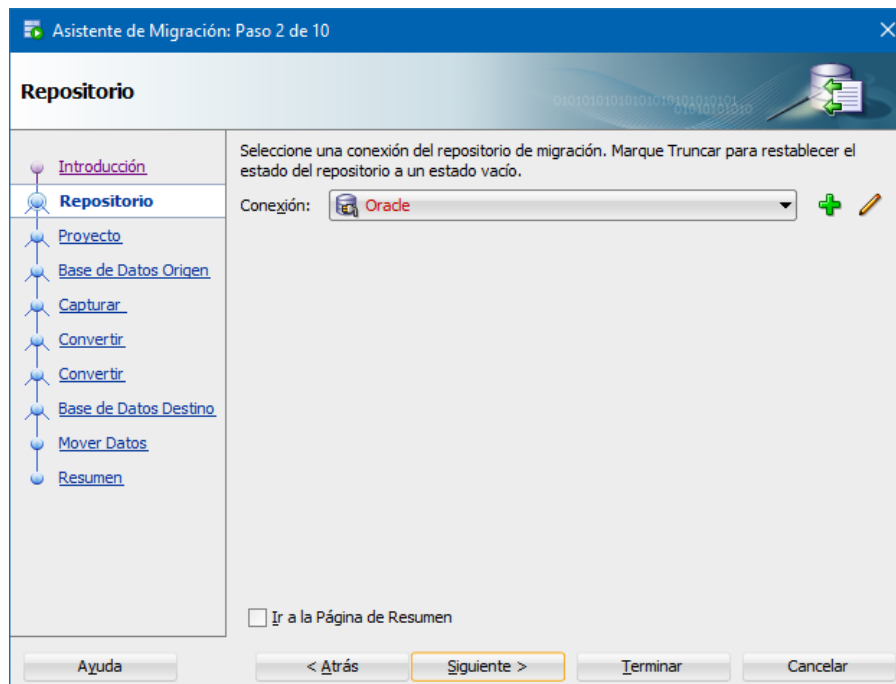


El asistente de migración consta de 10 pasos:

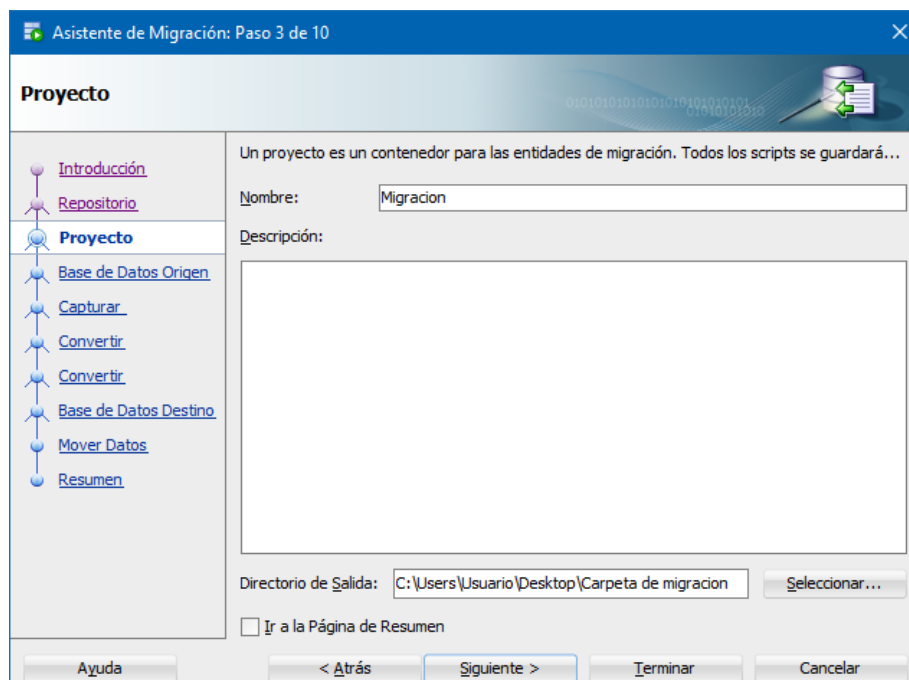
Paso1.- Descripción del procedimiento de migración.



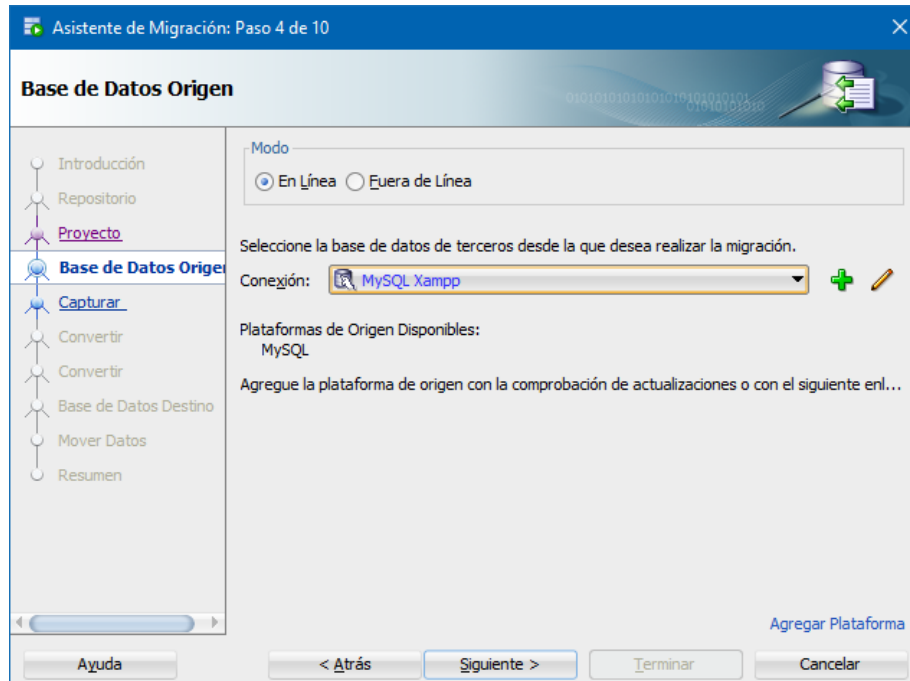
Paso2 .- Elegir la conexión de repositorio para almacenar el proceso de migración.



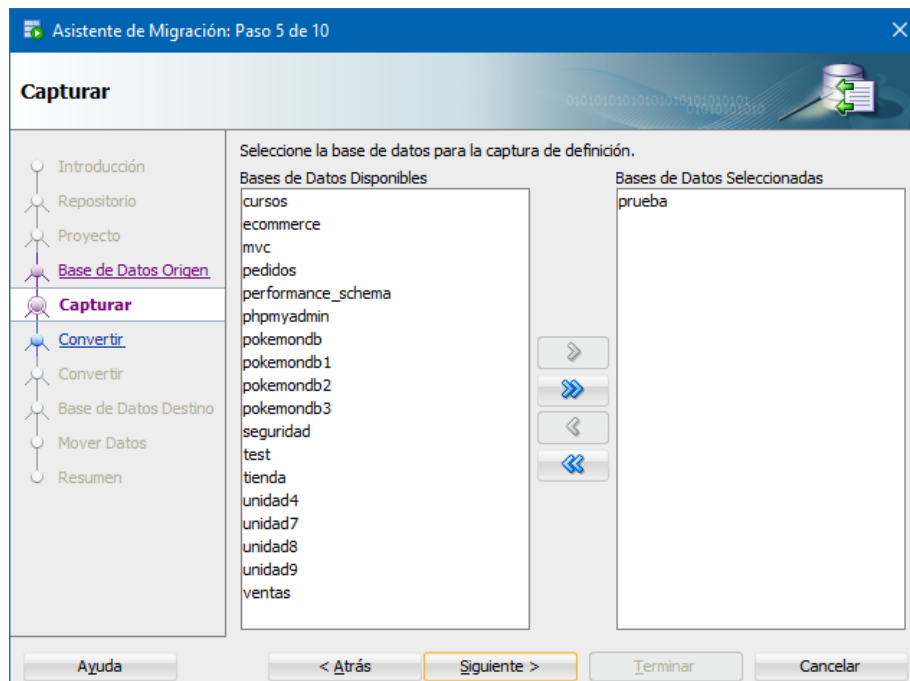
Paso3 .- Establecer un nombre del proceso de traspaso y un directorio para que se almacena los logs y scripts generados.



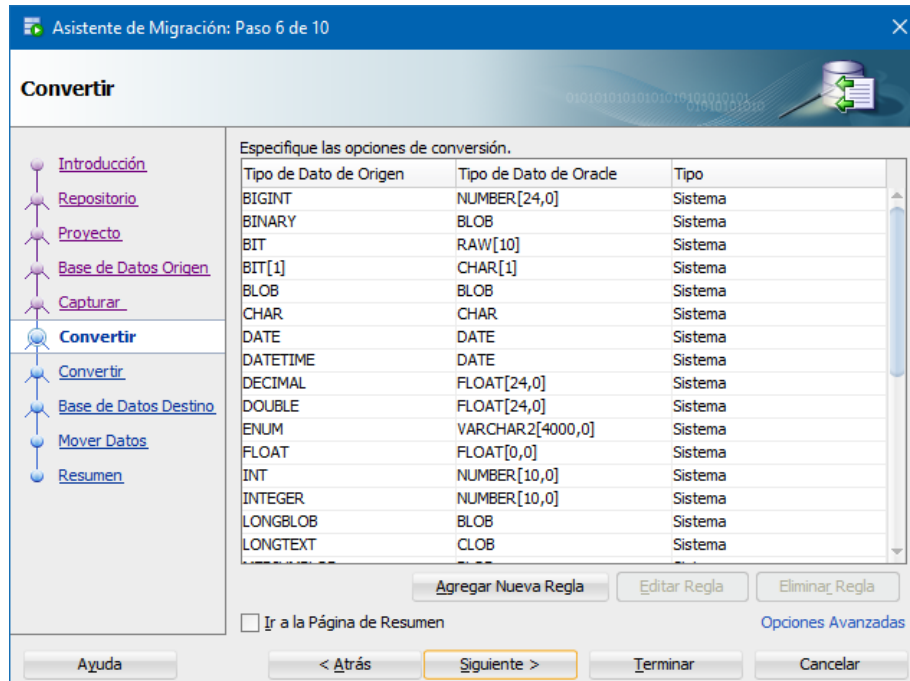
Paso4 .- Establecer que conexión de origen se iniciara el traspaso. En este caso es una base de datos MySQL.



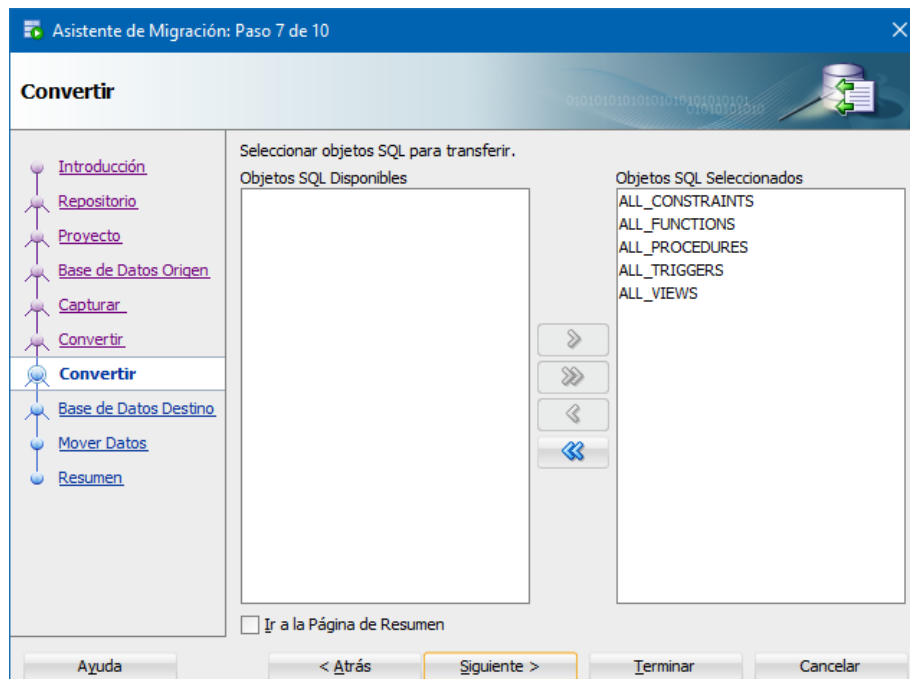
Paso5 .- Una vez indicado la conexión de origen se tendrá que seleccionar que base de datos se va a establecer, podemos elegir 1 o varias a la vez.



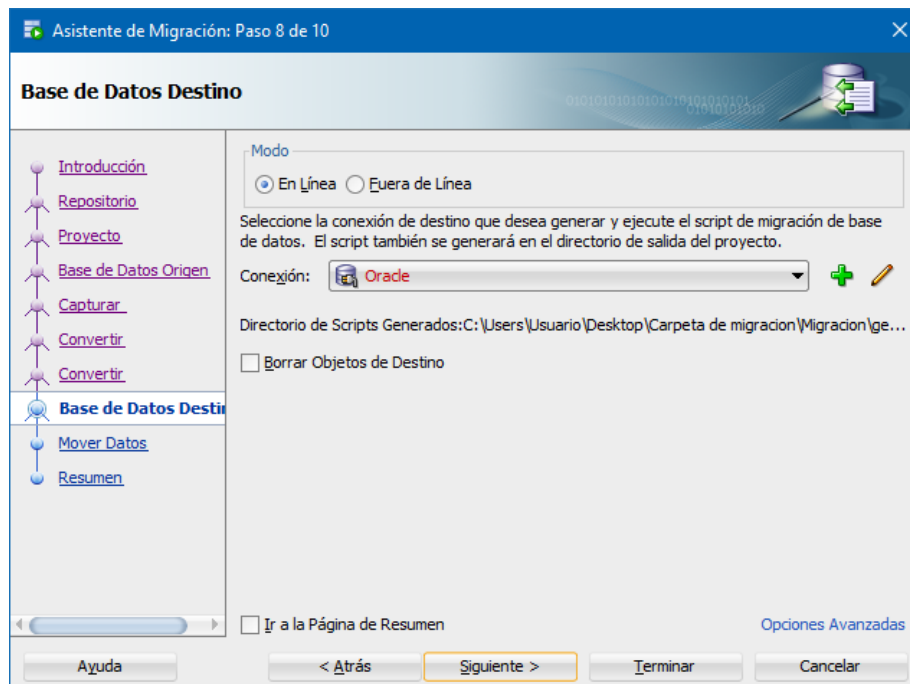
Paso6 .- Establecer los criterios de conversión, por ejemplo, los tipos de datos de origen a los tipos de datos destino. Es posible cambiar algún criterio o añadir nuevos. Se dejan por defecto.



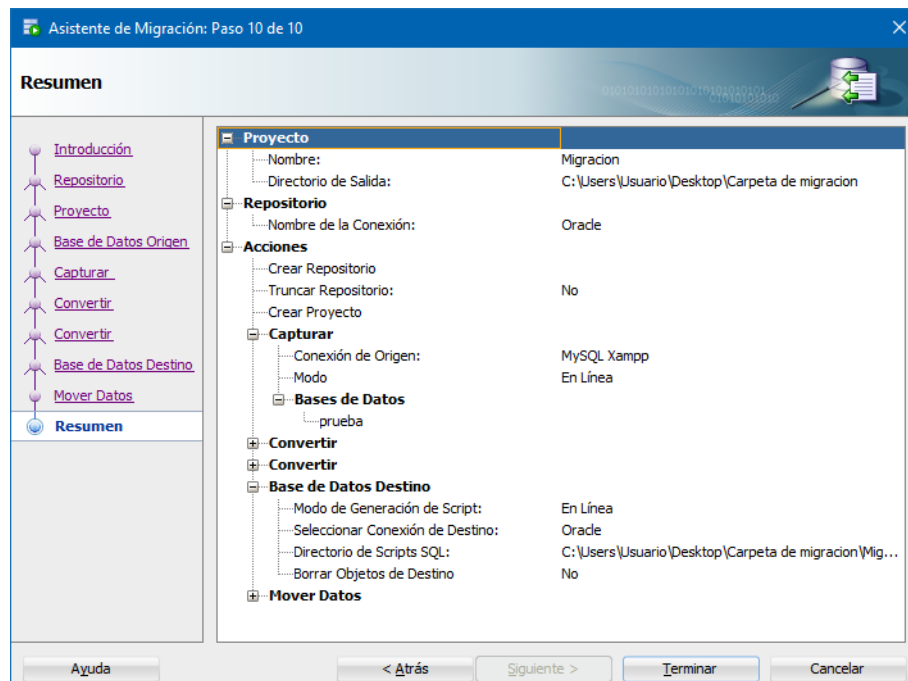
Paso7 .- Además de la estructura de la base de datos, es posible indicar que tipo de objetos podemos traspasar, relaciones, funciones, procedimientos, vistas, etc...



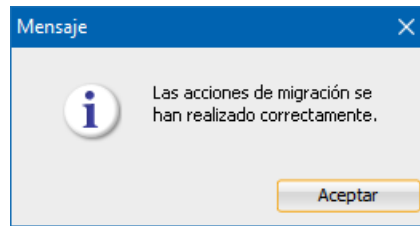
Paso8 .- Indicamos la conexión destino. En este caso, Oracle.



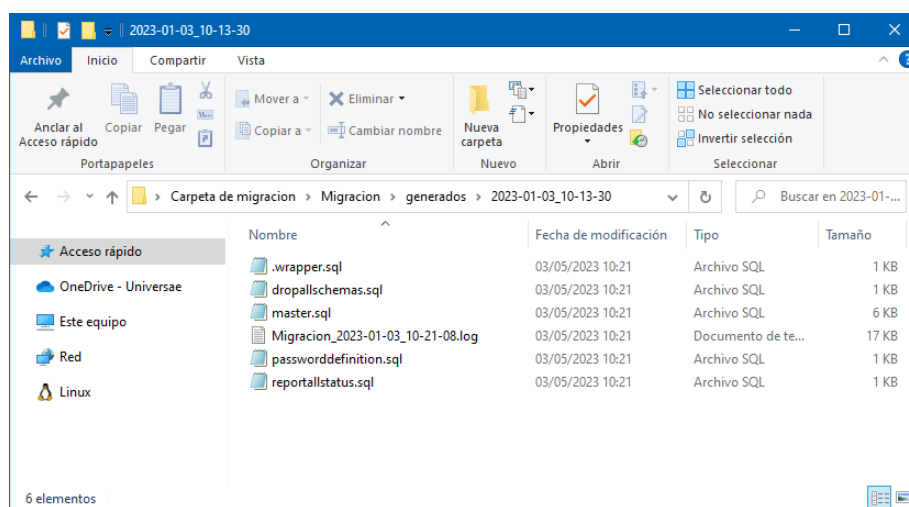
Paso9 y 10 .- Resumen de todo el proceso de traspaso de una base de datos MySQL a Oracle



Si todo ha ido bien, recibiremos un mensaje notificando la realización del proceso.



Verificamos los archivos generados en el directorio que habíamos especificado en el paso 3.



El fichero .log muestra todas las instrucciones ejecutadas en orden que han sido necesarias para realizar el proceso de migración.

```

C:\Users\Usuario\Desktop\Carpeta de migracion\Migracion\generados\2023-01-03_10-13-30\Migracion_2023-01-03_10-21-08.log - Notepa...
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Complementos  Pestañas  ?
Migracion_2023-01-03_10-21-08.log
25 del comando -
26 DROP ROLE ROLE_Migracion
27 Informe de error -
28 ORA-01919: el rol 'ROLE_MIGRACION' no existe
29 01919. 00000 - "role '%s' does not exist"
30 *Cause:   Role by that name does not exist.
31 *Action:  Verify you are using the correct role name.
32 SQL> PROMPT Creating Role ROLE_Migracion ...
33 Creating Role ROLE_Migracion ...
34 SQL> CREATE ROLE ROLE_Migracion ;
35
36 Error que empieza en la línea : 18 Archivo @ C:\Users\Usuario\Desktop\Carpeta de migracion\Migracion\
37 generados\2023-05-03_10-13-30\master.sql
38 del comando -
39 CREATE ROLE ROLE_Migracion
40 Informe de error -
41 ORA-65096: nombre de usuario o rol común no válido
42 65096. 00000 - "invalid common user or role name"
43 *Cause:   An attempt was made to create a common user or role with a name
44           that was not valid for common users or roles. In addition to the
45           usual rules for user and role names, common user and role names
46           must consist only of ASCII characters, and must contain the prefix
47           specified in common_user_prefix parameter.
48 *Action:  Specify a valid common user or role name.
49 SQL>
50 SQL> -- PROMPT Drop prueba user
51 SQL> -- drop user prueba cascade;
52 SQL>
53 SQL> PROMPT Create user prueba
54 Create user prueba
55 SQL> CREATE USER prueba IDENTIFIED BY &&prueba_password PASSWORD EXPIRE ACCOUNT LOCK /* DEFAULT
56           TABLESPACE USERS TEMPORARY TABLESPACE TEMP */;
57 Antiguo:CREATE USER prueba IDENTIFIED BY &&prueba_password PASSWORD EXPIRE ACCOUNT LOCK /* DEFAULT
58           TABLESPACE USERS TEMPORARY TABLESPACE TEMP */
59 Nuevo:CREATE USER prueba IDENTIFIED BY "NsB#8GCTQozfTulF" PASSWORD EXPIRE ACCOUNT LOCK /* DEFAULT
60           TABLESPACE USERS TEMPORARY TABLESPACE TEMP */
61
62 Error que empieza en la línea : 24 Archivo @ C:\Users\Usuario\Desktop\Carpeta de migracion\Migracion\
Structured Query Language length: 16.463  lines: 508  Ln: 43  Col: 76  Pos: 1.449  Unix (LF)  ANSI  INS

```

Finalmente, comprobamos que en la conexión destino aparecen toda la estructura de la base de datos igual que la base de datos origen. Es importante mencionar que no se han copiado los datos, solo la estructura, a partir de ahora, sería necesario exportar todas las sentencias DLM para replicar los datos hacia la base de datos Oracle.





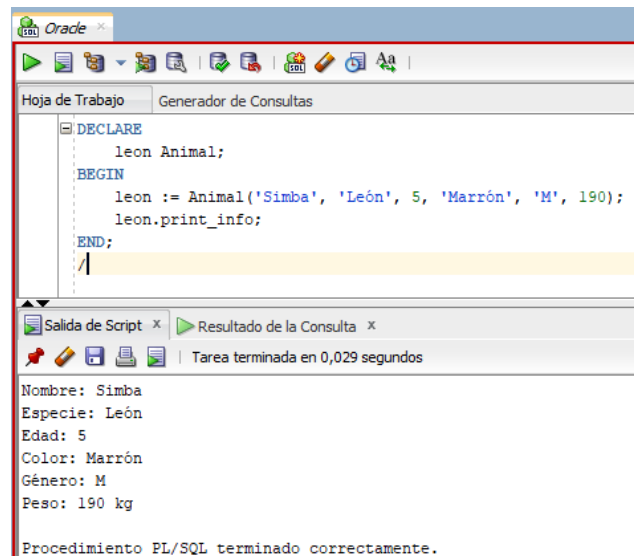
Unidad 8

Ejercicio 1 de la videoclase

Crear un objeto que represente un animal con los atributos que consideréis oportunos

```
-- Tipo Animal
CREATE OR REPLACE TYPE Animal AS OBJECT (
    nombre VARCHAR2(100),
    especie VARCHAR2(100),
    edad NUMBER,
    color VARCHAR2(100),
    genero CHAR(1), -- Atributo adicional: género (M o F)
    peso NUMBER,
    MEMBER PROCEDURE print_info
);
/
-- Cuerpo de la estructura de Animal
-- El procedimiento print_info muestra los datos de cada atributo
CREATE OR REPLACE TYPE BODY Animal AS
    MEMBER PROCEDURE print_info IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Nombre: ' || nombre);
        DBMS_OUTPUT.PUT_LINE('Especie: ' || especie);
        DBMS_OUTPUT.PUT_LINE('Edad: ' || TO_CHAR(edad));
        DBMS_OUTPUT.PUT_LINE('Color: ' || color);
        DBMS_OUTPUT.PUT_LINE('Género: ' || genero);
        DBMS_OUTPUT.PUT_LINE('Peso: ' || TO_CHAR(peso) || ' kg');
    END print_info;
END;
/

-- Uso del objeto Animal
DECLARE
    leon Animal;
BEGIN
    leon := Animal('Simba', 'León', 5, 'Marrón', 'M', 190);
    leon.print_info;
END;
/
```

```

DECLARE
    leon Animal;
BEGIN
    leon := Animal('Simba', 'León', 5, 'Marrón', 'M', 190);
    leon.print_info;
END;
/

```

Nombre: Simba
 Especie: León
 Edad: 5
 Color: Marrón
 Género: M
 Peso: 190 kg

Procedimiento PL/SQL terminado correctamente.

Ejercicio 2 de la videoclase

Crear diferentes tipos de animales (Marino, Terrestre, etc.) añadir los atributos que consideréis. Debéis tener en cuenta el ejercicio1.

Rectificamos Animales del ejercicio 1 para añadir NOT FINAL

```

-- Tipo Animal
CREATE OR REPLACE TYPE Animal AS OBJECT (
    nombre VARCHAR2(100),
    especie VARCHAR2(100),
    edad NUMBER,
    color VARCHAR2(100),
    genero CHAR(1),
    peso NUMBER,
    MEMBER PROCEDURE print_info
) NOT FINAL;
/

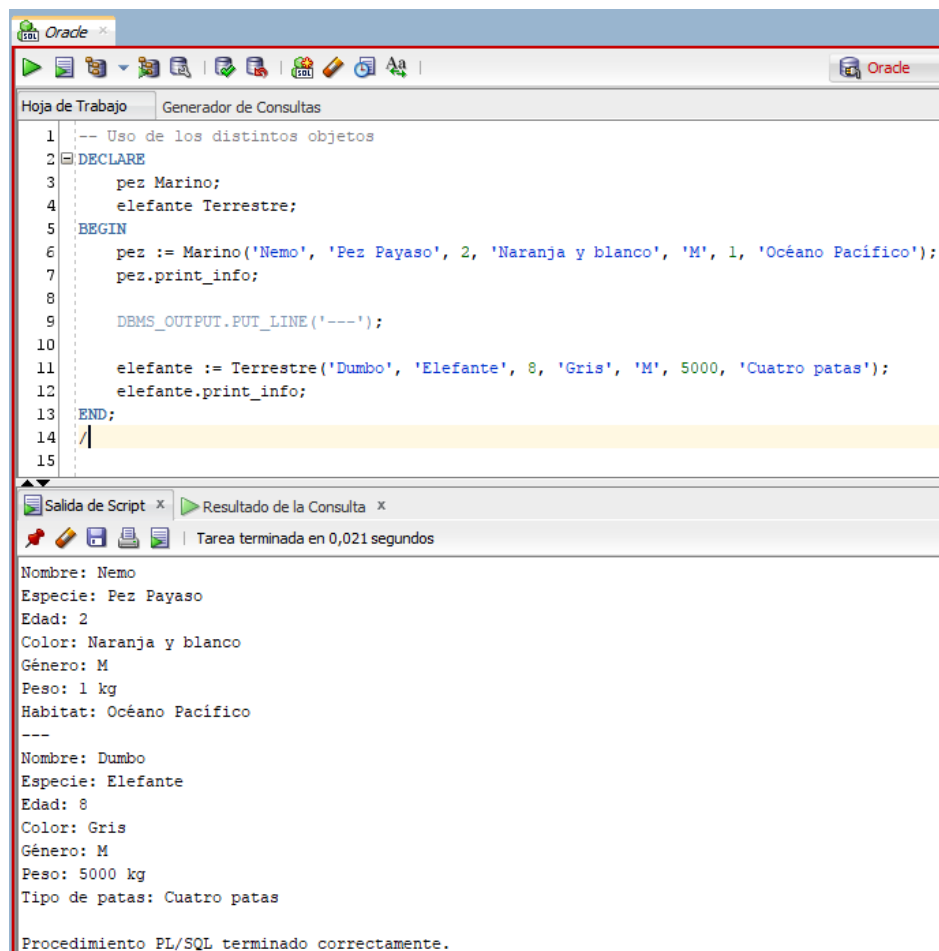
CREATE OR REPLACE TYPE BODY Animal AS
    MEMBER PROCEDURE print_info IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Nombre: ' || nombre);
        DBMS_OUTPUT.PUT_LINE('Especie: ' || especie);
        DBMS_OUTPUT.PUT_LINE('Edad: ' || TO_CHAR(edad));
        DBMS_OUTPUT.PUT_LINE('Color: ' || color);
        DBMS_OUTPUT.PUT_LINE('Género: ' || genero);
        DBMS_OUTPUT.PUT_LINE('Peso: ' || TO_CHAR(peso) || ' kg');
    END print_info;
END;
/

-- Tipo Marino, hereda de Animal

```



```
CREATE OR REPLACE TYPE Marino UNDER Animal (  
    habitat VARCHAR2(100),  
    OVERRIDING MEMBER PROCEDURE print_info  
);  
/  
  
CREATE OR REPLACE TYPE BODY Marino AS  
    OVERRIDING MEMBER PROCEDURE print_info IS  
    BEGIN  
        (SELF as Animal).print_info(); -- Llamar al procedimiento print_info de  
Animal  
        DBMS_OUTPUT.PUT_LINE('Habitat: ' || habitat);  
    END print_info;  
END;  
/  
  
-- Tipo Terrestre, hereda de Animal  
CREATE OR REPLACE TYPE Terrestre UNDER Animal (  
    tipo_patas VARCHAR2(100),  
    OVERRIDING MEMBER PROCEDURE print_info  
);  
/  
  
CREATE OR REPLACE TYPE BODY Terrestre AS  
    OVERRIDING MEMBER PROCEDURE print_info IS  
    BEGIN  
        (SELF as Animal).print_info(); -- Llamar al procedimiento print_info de  
Animal  
        DBMS_OUTPUT.PUT_LINE('Tipo de patas: ' || tipo_patas);  
    END print_info;  
END;  
/  
  
-- Uso de los distintos objetos  
DECLARE  
    pez Marino;  
    elefante Terrestre;  
BEGIN  
    pez := Marino('Nemo', 'Pez Payaso', 2, 'Naranja y blanco', 'M', 1, 'Océano  
Pacífico');  
    pez.print_info;  
  
    DBMS_OUTPUT.PUT_LINE('---');  
  
    elefante := Terrestre('Dumbo', 'Elefante', 8, 'Gris', 'M', 5000, 'Cuatro  
patas');  
    elefante.print_info;  
END;  
/
```



```

1  -- Uso de los distintos objetos
2  DECLARE
3      pez Marino;
4      elefante Terrestre;
5  BEGIN
6      pez := Marino('Nemo', 'Pez Payaso', 2, 'Naranja y blanco', 'M', 1, 'Océano Pacífico');
7      pez.print_info;
8
9      DBMS_OUTPUT.PUT_LINE('---');
10
11     elefante := Terrestre('Dumbo', 'Elefante', 8, 'Gris', 'M', 5000, 'Cuatro patas');
12     elefante.print_info;
13 END;
14 /
15
Salida de Script x Resultado de la Consulta x
Tarea terminada en 0,021 segundos

Nombre: Nemo
Especie: Pez Payaso
Edad: 2
Color: Naranja y blanco
Género: M
Peso: 1 kg
Habitat: Océano Pacífico
---
Nombre: Dumbo
Especie: Elefante
Edad: 8
Color: Gris
Género: M
Peso: 5000 kg
Tipo de patas: Cuatro patas

Procedimiento PL/SQL terminado correctamente.

```

Ejercicio 3 de la videoclase

Crear una tabla en la que se pueda tener un conjunto de animales de un tipo por columna. Por ejemplo, columna 1 = Marino, Columna 2 = Terrestre, etc.

Para almacenar un objeto por fila, separando cada columna como tipo de Animal

```

CREATE TABLE ANIMALES_OPCION1 (
    id NUMBER PRIMARY KEY,
    marino Marino,
    terrestre Terrestre
);

```

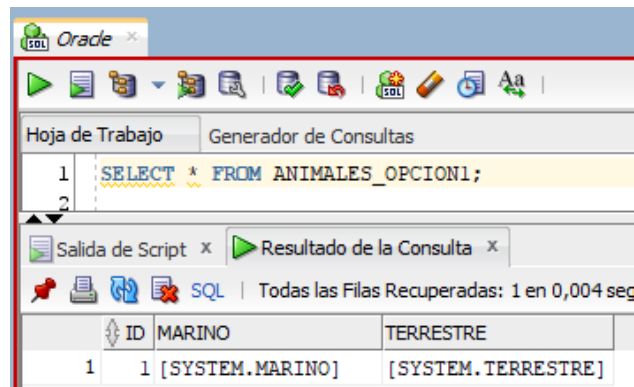
-- Insertar datos

```

INSERT INTO ANIMALES_OPCION1 VALUES (
    1,
    Marino('Nemo', 'Pez Payaso', 2, 'Naranja y blanco', 'M', 1, 'Océano
Pacífico'),
    Terrestre('Dumbo', 'Elefante', 8, 'Gris', 'M', 5000, 'Cuatro patas')
);

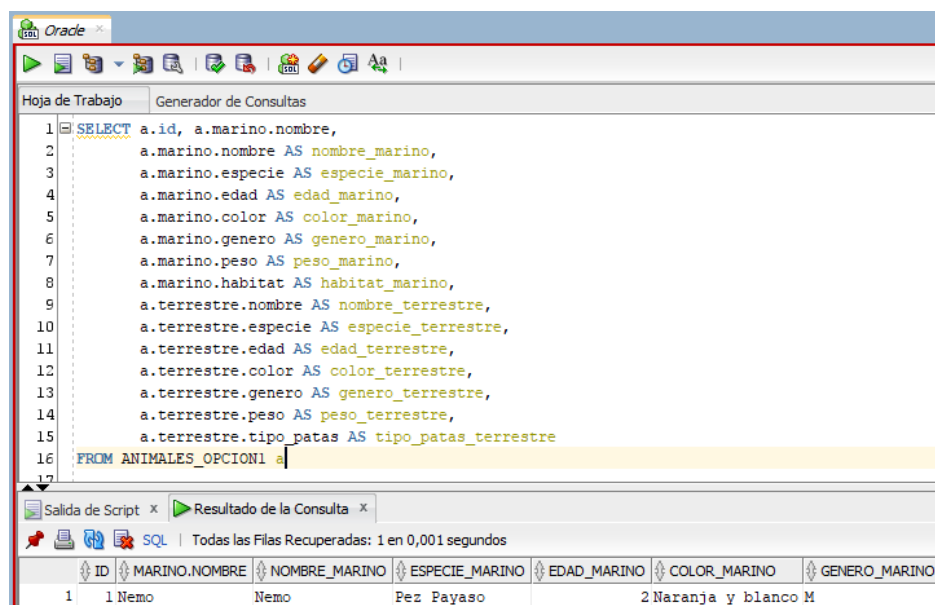
```

```
-- Mostrar datos
SELECT * FROM ANIMALES_OPCION1;
```



ID	MARINO	TERRESTRE
1	1 [SYSTEM.MARINO]	[SYSTEM.TERRESTRE]

```
SELECT a.id, a.marino.nombre,
       a.marino.nombre AS nombre_marino,
       a.marino.especie AS especie_marino,
       a.marino.edad AS edad_marino,
       a.marino.color AS color_marino,
       a.marino.genero AS genero_marino,
       a.marino.peso AS peso_marino,
       a.marino.habitat AS habitat_marino,
       a.terrestre.nombre AS nombre_terrestre,
       a.terrestre.especie AS especie_terrestre,
       a.terrestre.edad AS edad_terrestre,
       a.terrestre.color AS color_terrestre,
       a.terrestre.genero AS genero_terrestre,
       a.terrestre.peso AS peso_terrestre,
       a.terrestre.tipo_patas AS tipo_patas_terrestre
FROM ANIMALES_OPCION1 a
```



ID	MARINO.NOMBRE	NOMBRE_MARINO	ESPECIE_MARINO	EDAD_MARINO	COLOR_MARINO	GENERO_MARINO
1	1 Nemo	Nemo	Pez Payaso			

Para almacenar un conjunto de animales en cada fila, separando cada columna como tipo.

```
CREATE TYPE Marino_tabla AS TABLE OF Marino;
/
CREATE TYPE Terrestre_tabla AS TABLE OF Terrestre;
/
CREATE TABLE ANIMALES_OPCION2 (
    id NUMBER PRIMARY KEY,
    marinos Marino_tabla,
    terrestres Terrestre_tabla
) NESTED TABLE marinos STORE AS marinos_nt,
  NESTED TABLE terrestres STORE AS terrestres_nt;

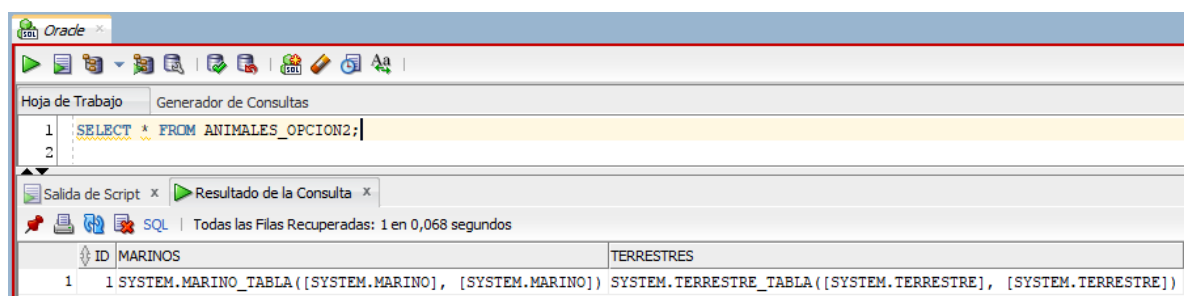
-- Insertar datos
DECLARE
    marinos_tab Marino_tabla;
    terrestres_tab Terrestre_tabla;
BEGIN
    marinos_tab := Marino_tabla(
        Marino('Nemo', 'Pez Payaso', 2, 'Naranja y blanco', 'M', 1, 'Océano
Pacífico'),
        Marino('Dory', 'Pez Cirujano', 3, 'Azul y amarillo', 'F', 1, 'Gran Barrera
de Coral')
    );

    terrestres_tab := Terrestre_tabla(
        Terrestre('Dumbo', 'Elefante', 8, 'Gris', 'M', 5000, 'Cuatro patas'),
        Terrestre('Simba', 'León', 5, 'Amarillo', 'M', 190, 'Cuatro patas')
    );

    INSERT INTO ANIMALES_OPCION2 VALUES (1, marinos_tab, terrestres_tab);

    COMMIT;
END;
/

-- Mostrar datos
SELECT * FROM ANIMALES_OPCION2
```

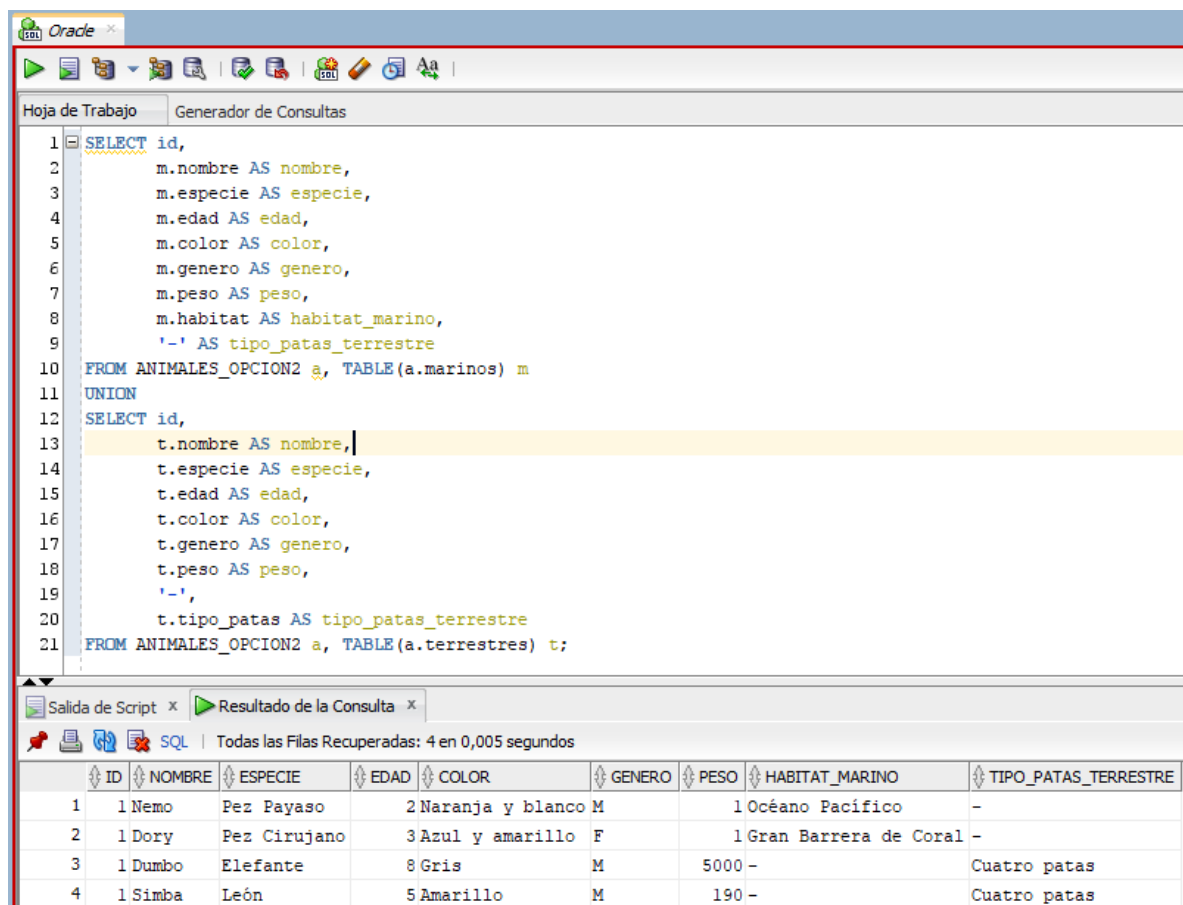


ID	MARINOS	TERRESTRES
1	1 SYSTEM.MARINO_TABLA([SYSTEM.MARINO], [SYSTEM.MARINO])	SYSTEM.TERRESTRE_TABLA([SYSTEM.TERRESTRE], [SYSTEM.TERRESTRE])

```

SELECT id,
       m.nombre AS nombre,
       m.especie AS especie,
       m.edad AS edad,
       m.color AS color,
       m.genero AS genero,
       m.peso AS peso,
       m.habitat AS habitat_marino,
       '-' AS tipo_patas_terrestre
FROM ANIMALES_OPCION2 a, TABLE(a.marinosa) m
UNION
SELECT id,
       t.nombre AS nombre,
       t.especie AS especie,
       t.edad AS edad,
       t.color AS color,
       t.genero AS genero,
       t.peso AS peso,
       '-',
       t.tipo_patas AS tipo_patas_terrestre
FROM ANIMALES_OPCION2 a, TABLE(a.terrestres) t;

```



The screenshot shows the Oracle SQL Developer interface. The top pane displays the SQL query, and the bottom pane shows the results of the query in a table format.

SQL Query:

```

SELECT id,
       m.nombre AS nombre,
       m.especie AS especie,
       m.edad AS edad,
       m.color AS color,
       m.genero AS genero,
       m.peso AS peso,
       m.habitat AS habitat_marino,
       '-' AS tipo_patas_terrestre
FROM ANIMALES_OPCION2 a, TABLE(a.marinosa) m
UNION
SELECT id,
       t.nombre AS nombre,
       t.especie AS especie,
       t.edad AS edad,
       t.color AS color,
       t.genero AS genero,
       t.peso AS peso,
       '-',
       t.tipo_patas AS tipo_patas_terrestre
FROM ANIMALES_OPCION2 a, TABLE(a.terrestres) t;

```

Query Results:

ID	NOMBRE	ESPECIE	EDAD	COLOR	GENERO	PESO	HABITAT_MARINO	TIPO_PATAS_TERRESTRE
1	Nemo	Pez Payaso	2	Naranja y blanco	M	1	Océano Pacífico	-
2	Dory	Pez Cirujano	3	Azul y amarillo	F	1	Gran Barrera de Coral	-
3	Dumbo	Elefante	8	Gris	M	5000	-	Cuatro patas
4	Simba	León	5	Amarillo	M	190	-	Cuatro patas



Ejercicio 4 de la videoclase

Modifica el objeto animal para que realice un procedimiento llamado ataque y muestre por mensaje si el animal puede atacar o no.

```
CREATE OR REPLACE TYPE Animal AS OBJECT (
    nombre VARCHAR2(100),
    especie VARCHAR2(100),
    edad NUMBER,
    color VARCHAR2(100),
    genero CHAR(1),
    peso NUMBER,
    puede_atacar CHAR(1), -- Añado un atributo puede_atacar para verificar si el
animal puede atacar
    MEMBER PROCEDURE print_info,
    MEMBER PROCEDURE ataque -- Añado el nuevo procedimiento ataque
) NOT FINAL;
/

CREATE OR REPLACE TYPE BODY Animal AS
    MEMBER PROCEDURE print_info IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Nombre: ' || nombre);
        DBMS_OUTPUT.PUT_LINE('Especie: ' || especie);
        DBMS_OUTPUT.PUT_LINE('Edad: ' || TO_CHAR(edad));
        DBMS_OUTPUT.PUT_LINE('Color: ' || color);
        DBMS_OUTPUT.PUT_LINE('Género: ' || genero);
        DBMS_OUTPUT.PUT_LINE('Peso: ' || TO_CHAR(peso) || ' kg');
    END print_info;

    MEMBER PROCEDURE ataque IS -- Implemento el procedimiento ataque
    BEGIN
        IF puede_atacar = 'S' THEN
            DBMS_OUTPUT.PUT_LINE(nombre || ' puede atacar.');
```

```
        ELSE
            DBMS_OUTPUT.PUT_LINE(nombre || ' no puede atacar.');
```

```
        END IF;
    END ataque;
END;
/

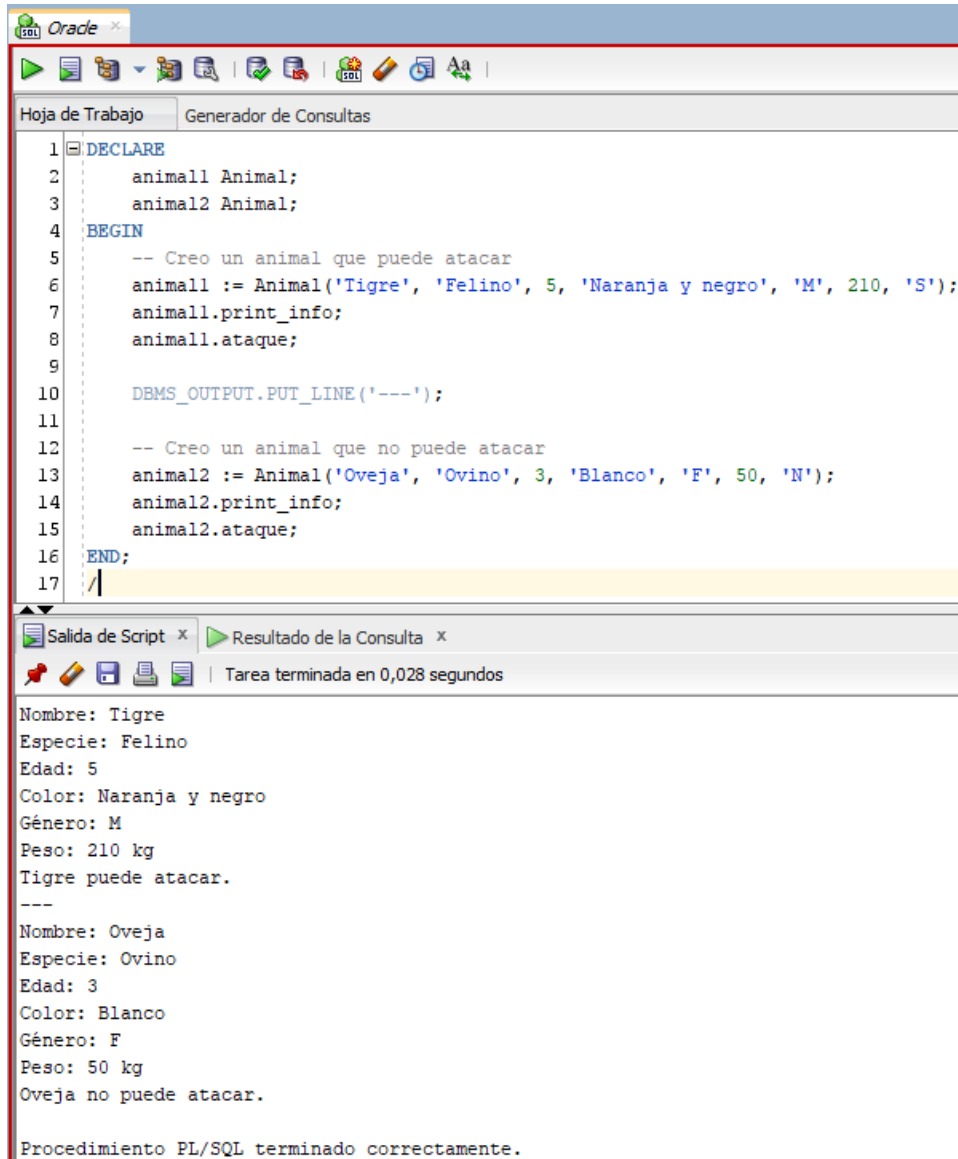
-- Ejemplo de uso
DECLARE
    animal1 Animal;
    animal2 Animal;
BEGIN
    -- Creo un animal que puede atacar
    animal1 := Animal('Tigre', 'Felino', 5, 'Naranja y negro', 'M', 210, 'S');
```

```
    animal1.print_info;
    animal1.ataque;
```

```
END;
```

```
DBMS_OUTPUT.PUT_LINE('---');
```

```
-- Creo un animal que no puede atacar  
animal2 := Animal('Oveja', 'Ovino', 3, 'Blanco', 'F', 50, 'N');  
animal2.print_info;  
animal2.ataque;  
END;  
/
```



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script declares two variables, `animal1` and `animal2`, of type `Animal`. It then creates `animal1` as a Tiger (Tigre) and `animal2` as a Sheep (Oveja). It prints information for both and attempts to call the `ataque` method on each. The bottom pane, titled 'Salida de Script' and 'Resultado de la Consulta', shows the output of the script. It displays the details for the Tiger, followed by a separator line (---), and then the details for the Sheep. The script concludes with the message 'Procedimiento PL/SQL terminado correctamente.'

```
1 DECLARE  
2     animal1 Animal;  
3     animal2 Animal;  
4 BEGIN  
5     -- Creo un animal que puede atacar  
6     animal1 := Animal('Tigre', 'Felino', 5, 'Naranja y negro', 'M', 210, 'S');  
7     animal1.print_info;  
8     animal1.ataque;  
9  
10    DBMS_OUTPUT.PUT_LINE('---');  
11  
12    -- Creo un animal que no puede atacar  
13    animal2 := Animal('Oveja', 'Ovino', 3, 'Blanco', 'F', 50, 'N');  
14    animal2.print_info;  
15    animal2.ataque;  
16 END;  
17 /
```

Nombre: Tigre
Especie: Felino
Edad: 5
Color: Naranja y negro
Género: M
Peso: 210 kg
Tigre puede atacar.

Nombre: Oveja
Especie: Ovino
Edad: 3
Color: Blanco
Género: F
Peso: 50 kg
Oveja no puede atacar.

Procedimiento PL/SQL terminado correctamente.

UNIVERSAE

— CHANGE YOUR WAY —

