

ÍNDICE:

ESCENARIO.....	4
INTRODUCCIÓN.....	5
1º PARTE:.....	6
P1 Comparar los diferentes tipos de modelos de bases de datos:.....	7
M1 Evaluar cómo se relacionan modelos de bases de datos y el proceso de normalización.....	12
D1 Evaluar diferentes Sistemas de gestión de bases de datos:.....	14
2º PARTE:.....	17
LO2 Diseñar un sistema de gestión de bases de datos utilizando un modelo relacional para satisfacer los requisitos del cliente.....	17
P2 Producir un diseño para un sistema de gestión de base de datos relacional para cumplir con los requisitos del cliente.....	19
M2 Analizar cómo el diseño optimizará el rendimiento del sistema.....	29
3º PARTE:.....	36
LO3 Desarrollar un sistema de gestión de bases de datos utilizando una plataforma adecuada.....	36
P4 Pruebe la funcionalidad y el rendimiento del sistema.....	41
M3 Implementar funciones eficaces en la solución para gestionar la concurrencia, la seguridad, las autorizaciones de usuarios y la recuperación de datos.....	49
D2 Evaluar la eficacia del diseño y desarrollo del sistema con respecto a los requisitos del cliente y del sistema.....	49
4º PARTE:.....	50
LO4 Demostrar las herramientas de administración y gestión de sistemas disponibles en la plataforma elegida.....	51
P6 Demostrar las herramientas disponibles en el sistema para gestionar la seguridad y las autorizaciones.....	53
D3 Analizar cualquier mejora futura que pueda ser necesaria para garantizar la eficacia continuada del sistema de base de datos.....	57

ESCENARIO

Estudio de caso

Una empresa multinacional, en rápido crecimiento, ha llevado a cabo una investigación y ha descubierto que, con el paso de los años y a medida que la empresa seguía creciendo, sus numerosos sistemas de gestión de bases de datos se han vuelto dispares. La empresa se encuentra ahora utilizando muchas plataformas diferentes. La tecnología que soportaba su sistema se adquiere y añadía a la infraestructura en función de las necesidades, sin una planificación demasiado controlada. Los sistemas de gestión de bases de datos han prestado un buen servicio a la empresa, pero con el creciente uso de sistemas basados en la nube y el paso a los centros de datos, el director general desearía que usted:

(LO1). Prepáralos una presentación en la que se comparen y evalúen diferentes sistemas de gestión de bases de datos y sus sistemas operativos disponibles (tanto de código abierto como específicos de un proveedor), justificando los criterios de evaluación. Deberá abarcar los modelos de datos, los lenguajes de modelado y examinar específicamente las bases de datos relacionales y sus estructuras.

(LO2 & LO3). Diseñe y desarrolle un sistema de gestión de bases de datos relacionales para la empresa y evalúe críticamente en función de los requisitos del cliente y del sistema.

El diseño deberá tener en cuenta al personal clave implicado en los sistemas de gestión de bases de datos, sus funciones y responsabilidades, y las características de los sistemas que deben gestionarse, como el almacenamiento de datos, las copias de seguridad, la seguridad, la recuperación y la concurrencia. Indique claramente cuáles serán los criterios para determinar el desarrollo de cualquier nuevo sistema de gestión de bases de datos y muestre cómo el diseño optimizará el rendimiento del sistema.

Utilizando un lenguaje de código abierto, desarrolle una aplicación totalmente funcional para interactuar con la base de datos. Debe incluir una interfaz de usuario y los resultados adecuados. Rellene la base de datos con los datos de prueba adecuados y pruebe completamente el sistema utilizando las pruebas apropiadas, como la seguridad, la integridad referencial y la funcionalidad, y utilice los resultados de las pruebas para optimizarlo. Redacte un informe sobre su base de datos, que incluya el diseño, el desarrollo y las pruebas, y evalúe críticamente el sistema en relación con los requisitos del cliente y del sistema.

(LO4). Demuestre su sistema de base de datos al Consejo de Administración.

El Director de TI está especialmente interesado en cómo supervisar y optimizar el rendimiento del sistema, así como en acceder a los registros de auditoría, y al Director de Cumplimiento le gustaría ver cómo se gestionan la seguridad y las autorizaciones, especialmente con respecto al GDPR. Evaluar las herramientas de administración y mantenimiento utilizadas e identificar las áreas susceptibles de mejora y debatir posibles mejoras adicionales para mantener el sistema "a prueba de futuro".

INTRODUCCIÓN

En respuesta al crecimiento acelerado de una empresa multinacional, se ha identificado la necesidad de abordar las diversidades en sus sistemas de gestión de bases de datos. La falta de planificación en la adopción de tecnologías ha resultado en una variedad de plataformas, generando la necesidad de una evaluación integral y una solución coherente.

Este estudio se enfoca en cuatro objetivos clave: *la comparación de sistemas de gestión de bases de datos, el diseño y desarrollo de un sistema relacional, la creación de una aplicación funcional y la presentación del sistema al Consejo de Administración*. Se abordarán modelos de datos, lenguajes de modelado y aspectos clave como seguridad y rendimiento.

El resultado final no solo busca resolver los desafíos actuales, sino también establecer bases sólidas para la sostenibilidad y adaptabilidad futuras. La implementación se guiará por una evaluación crítica, asegurando que el sistema cumpla con los requisitos del cliente y del sistema en constante evolución.

1º PARTE:

P1 Comparar los diferentes tipos de modelos de bases de datos.

Explica los distintos modelos de bases de datos. Un ejemplo de fuente de información podría ser: <https://ayudaleyprotecciondatos.es/bases-de-datos/modelos/>

M1 Evaluar cómo se relacionan modelos de bases de datos y el proceso de normalización puede proporcionar estructuras de datos eficientes.

Céntrate en el modelo relacional. Explica cómo se puede crear una estructura eficiente. Repasa el proceso de diseño de una BD (diseños conceptual, lógico y físico). Explica el proceso de normalización.

D1 Evaluar diferentes Sistemas de gestión de bases de datos en relación con el código abierto y específicos del proveedor de plataformas, justificando los criterios utilizados en la evaluación.

Enumera varios productos DBMS e indica sus características (si son de código abierto o propiedad de un proveedor, modelo de datos utilizado, así como otras características que consideres importantes).

Para esta primera parte del HITO os puede ayudar la revisión de los apuntes del año pasado de la asignatura de Bases de datos.

BASES DE DATOS:

Los modelos de bases de datos son representaciones “abstractas” de cómo se estructuran y organizan los datos en una base de datos. Estas representaciones definen la forma en que los datos se almacenan, de cómo se acceden y gestionan dentro de un sistema de gestión de bases de datos, o también conocido por sus siglas DBMS.

Los modelos de bases de datos proporcionan a los diseñadores y desarrolladores de bases de datos comprender y trabajar con la información de manera coherente.

Las bases de datos se utilizan en una amplia variedad de aplicaciones y entornos.

Algunos ejemplos son:

- *Gestión de Clientes*
- *Gestión de Inventarios*
- *Gestión de Recursos Humanos*
- *Reservas de viajes*
- *Aplicaciones de Redes Sociales*
- *Comercio Electrónico*
- *información Geográfica*
- *Sistemas de Salud.*
- *Banca y Finanzas*

P1 Comparar los diferentes tipos de modelos de bases de datos:

Actualmente existen varios modelos de bases de datos, y cada uno de ellos tiene diferentes requisitos, que adaptas a tus necesidades.

Algunos modelos de bases de datos más comunes son:

1. **Modelo Relacional:** El modelo relacional es uno de los más comunes y más utilizados. En este modelo, los datos se organizan en tablas que constan de filas y columnas. Cada columna representa un atributo de la entidad (como nombre, dirección, fecha de nacimiento), y cada fila contiene datos sobre una instancia de la entidad.

Ejemplo de aplicaciones:

- incluyen MySQL y PostgreSQL

Se basan en la álgebra relacional y es eficaz para estructurar datos en una variedad de aplicaciones.

2. **Modelo de Bases de Datos NoSQL:** Este modelo permite el almacenamiento y recuperación de datos no estructurados o semiestructurados. Se diferencia del modelo relacional, ya que NO se utilizan tablas rígidas.

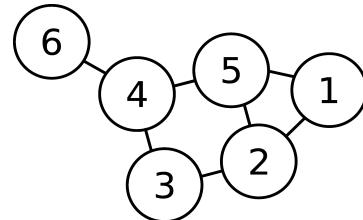
Ejemplos:

- bbdd de documentos (como MongoDB)
- bbdd de grafos (como Neo4j)
- bbdd de clave-valor (como Redis).

Modelos ideales para aplicaciones que manejan datos altamente variables y no requieren una estructura fija.

Grafo ⇒ Se utilizan para representar relaciones entre objetos o entidades. Estructura matemática con nodos y aristas que conectan los nodos.

- Nodos (vértices) → Representan entidades individuales, como ciudades en un mapa o personas en Instagram, básicamente son elementos de un conjunto. Y cada nodo puede llevar información adicional.
- Aristas (bordes) → Representan las relaciones entre los nodos. Pueden ser unidireccionales o bidireccionales.



3. Modelo Jerárquico: Este modelo se utiliza con menos frecuencia actualmente.

Organiza los datos en una estructura de árbol invertido. Cada registro tiene un nodo raíz, y los nodos secundarios surgen de este nodo raíz. Cada nodo padre puede tener múltiples nodos hijos, pero cada nodo hijo solo puede tener un nodo padre, como un árbol invertido.

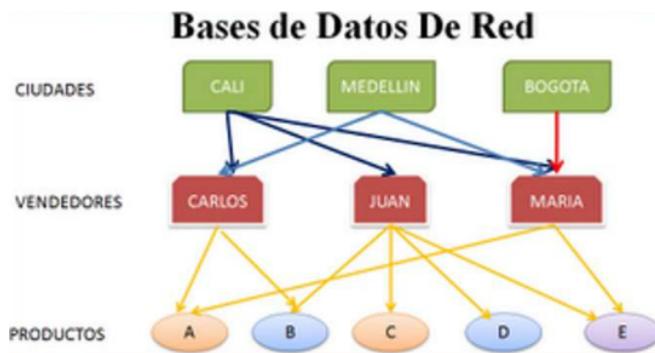
Ejemplo:

- Sistema IBM Information Management System (IMS).

MODELO DE DATOS JERARQUICO



4. Modelo de Red: Parecido al modelo jerárquico, pero permite relaciones más complejas entre registros. Los registros principales pueden tener múltiples relaciones con otros registros, lo que lo hace que sea más flexible en términos de representar relaciones de datos complejas.



Ejemplo:

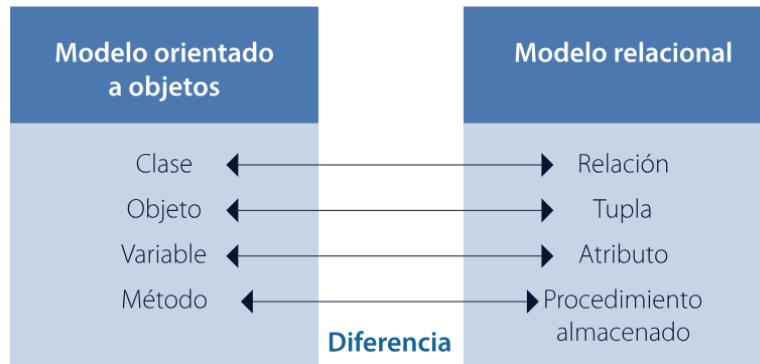
- Integrated Data Store (IDS).

5. Modelo de Bases de Datos Orientado a Objetos: En este modelo, los datos se almacenan en forma de objetos, facilitando la representación de datos.

Ejemplo:

- Zope Object Database (ZODB).

Es útil para aplicaciones que utilizan lenguajes de programación orientados a objetos como Java.



Modelo de Bases de Datos Temporales: Este modelo se centra en el cambio en los datos a lo largo del tiempo. Se utiliza en aplicaciones que requieren la gestión de versiones de datos a lo largo del tiempo.

Ejemplo:

- Oracle Workspace Manager



6. Modelo de Bases de Datos Multidimensional: Este modelo se utiliza principalmente en aplicaciones OLAP, procesamiento analítico en línea. Realiza consultas y análisis detallados de datos. Los datos se organizan en una estructura de cubo multidimensional.

Es eficiente para la recuperación de datos, pero puede ser difícil de modificar una vez creada.

Un ejemplo: Tienda.

Con sus Dimensiones:

- Tiempo = Trimestre y Año
- Producto = Categoría y nombre
- Ubicación = Tienda y Ciudad
- Cliente = nombre y ID

Y sus Medidas son:

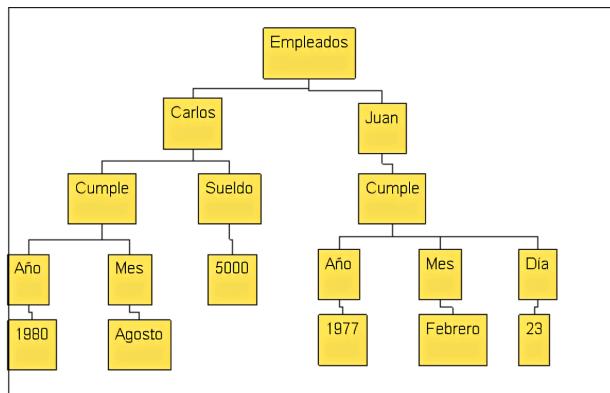
- Ventas
- Ganancias
- Descuentos

	Abril			Mayo			Junio		
	Argentina	Brasil	Chile	Argentina	Brasil	Chile	Argentina	Brasil	Chile
Producto1	212	534	254						
Producto2	21	46	33						
Producto3	310	321	200						
Producto4	120	234	131						
Producto5	43	78	55						
Producto6	12	32	21						

7. Modelo de Bases de Datos Semiestructurado: Se centra en los datos que no siguen una estructura tabular rígida (se representa como el modelo relacional, en forma de tabla), como los datos de una página web. Los datos se organizan en una estructura jerárquica y se pueden agrupar utilizando etiquetas.

Ejemplo:

- Catálogos de productos y tiendas en línea
- bioinformática
- Datos geoespaciales



8. Modelo de Bases de Datos Asociativo: Divide los datos en *entidades* y *asociaciones*.

Las *entidades* ⇒ objetos independientes

Las *asociaciones* ⇒ representan relaciones entre estas entidades.

Se utiliza, sobre todo, para datos en los que existen relaciones complejas.

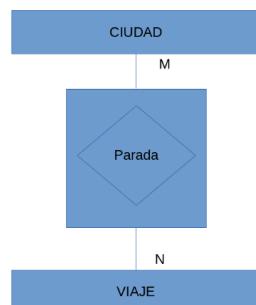
Ejemplo:

- En la redes sociales

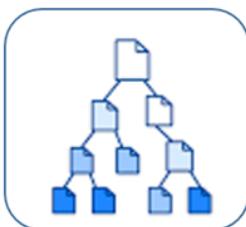
9. Modelos de Bases de Datos NoSQL (Submodelos): Los sistemas de bases de datos

NoSQL incluyen **submodelos** como:

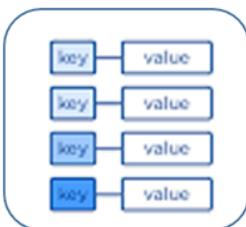
- modelo de base de datos gráfico
- modelo de base de datos multivalor
- modelo de base de datos de documentos
- modelo de base de datos de columnas



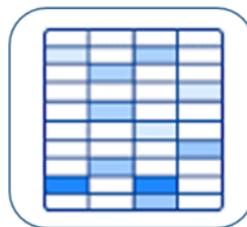
Estos submodelos se adaptan a las necesidades específicas de almacenamiento y recuperación de datos.



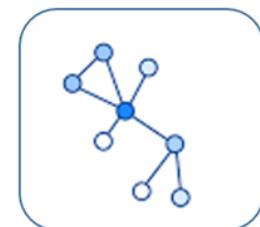
Document
Store



Key-Value
Store



Wide-Column
Store



Graph
Store

Tu elección del modelo de base de datos, depende de tus necesidades, preferencias y de los requisitos de los datos que se van a almacenar y gestionar. Obviamente cada modelo tiene sus propias ventajas y desventajas y deben ser elegidos para ciertos tipos de aplicaciones y casos de uso.

M1 Evaluar cómo se relacionan modelos de bases de datos y el proceso de normalización.

Modelo Relacional:

El proceso de normalización es fundamental en el *modelo relacional* para crear *estructuras de datos eficientes*, organiza y estructura una base de datos de manera eficiente para reducir la redundancia de datos y evitar problemas de integridad. La **normalización** es, básicamente, la división de una tabla en múltiples tablas más pequeñas para eliminar las partes repetidas e inútiles de los datos y mantener la integridad de los mismos, de manera consistente en toda la base de datos. Como todo, va por etapas, llamadas *Formas Normales (FN)*, pautas para organizar los datos de una base de datos, asegurándose de que los datos estén organizados de la manera más eficiente posible:

1. **Primera Forma Normal (1FN):** En esta etapa, se busca eliminar la redundancia de datos a nivel de columna, asegurándose de que cada columna de la tabla tenga valores que no se pueden dividir en partes más pequeñas(valores atomicos).
Un ejemplo: Tabla de "Habitantes" donde cada columna tiene información única sobre cada persona, sin columnas repetidas.
2. **Segunda Forma Normal (2FN):** Después de cumplir con 1FN, en esta etapa, se busca eliminar la redundancia de datos a nivel de fila. Además, todos los atributos *NO CLAVE* deben depender de la *CLAVE PRINCIPAL*. Es decir, cada columna *no clave* debe estar relacionada con la *clave principal*.
Por ejemplo: Tabla "Pedidos" con productos, la información del producto debe vincularse al pedido, no redundante en cada fila de pedido.

3. **Tercera Forma Normal (3FN):** Despues de cumplir con 1FN y 2FN, se busca eliminar la dependencia transitiva (*relaciones en una base de datos donde un elemento A depende de otro elemento B, y este último depende de un tercer elemento C*) entre columnas *no clave*. Es decir, si una columna *no clave* depende de otra columna *no clave*, esa dependencia se eliminará.

Por ejemplo: Tabla "Empleados" con información sobre el departamento y el gerente, la información del departamento debe depender de la clave principal de empleado, no del gerente.

EJEMPLO: Tenemos una tabla llamada "*Facturas*", con columnas:

- ID de factura
- Nombre del cliente
- Nombre del vendedor

Para lograr normalizarla, podríamos dividirla en tres tablas:

- "*Facturas*" (con ID de factura)
- "*Clientes*" (con detalles del cliente)
- "*Vendedores*" (con detalles del vendedor).

Aquí cumple con la 1FN al tener *valores atómicos* y con la 2FN al eliminar la redundancia de datos en columnas *no clave*.

Para cumplir con la 3FN, nos debemos asegurar que la tabla de "*Clientes*" y "*Vendedores*" no tengan dependencias transitivas.

La normalización, como ya hemos dicho, es un proceso importante en el diseño de bases de datos para garantizar la eficiencia y la integridad de los datos. Luego estarían las formas normales superiores (4FN, 5FN), que también existen pero para abordar situaciones más complejas, pero todavía no las utilizaremos.

El diseño de una base de datos eficiente implica:

- 1º Identificar las entidades y relaciones clave en el sistema.
- 2º Diseñar un modelo conceptual que represente estas entidades y relaciones.
- 3º Llevar el modelo conceptual al modelo lógico, que incluye la creación de tablas y definición de claves primarias y foráneas.
- 4º Aplicar el proceso de *normalización* para eliminar la redundancia y reducir el riesgo de errores de los datos.
- 5º Finalmente, crear el modelo físico que define cómo se implementarán las tablas en el sistema de gestión de bases de datos.

La normalización es esencial para garantizar la integridad de los datos y reducir el espacio de almacenamiento necesario.

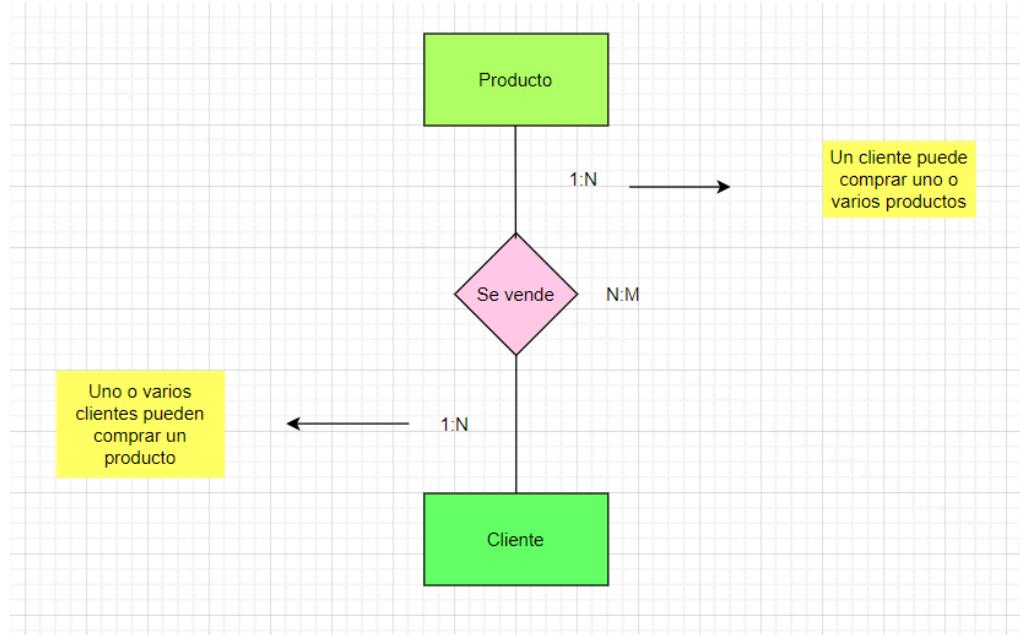
Ahora lo pasaremos a la realidad con un ejemplo simple y corto, para verlo más claro:

1º Tendremos que saber de qué va a ir nuestra base de datos. Haremos una tienda:

Entidades ⇒ Producto - Cliente

Relaciones ⇒ Se vende uno o varios productos a un cliente

2º Hacemos un modelo conceptual:



3º Bueno aqui pasaremos del modelo conceptual al lógico, que es hacerlo mas específico, poniendo todo lo que queremos, con más detalles, además de presentación de las claves primarias y foráneas.

4º Ahora aplicamos la normalización utilizando 1NF, 2NF, 3NF, etc... Son los tipos de normalización.

5º Aquí ya crearemos el modelo con bbdd en la aplicación que deseas.

D1 Evaluar diferentes Sistemas de gestión de bases de datos:

Para empezar analicemos que es un sistema de gestión:

Un Sistema de Gestión de Bases de Datos(DBMS) es un software o sistema que permite crear, gestionar, administrar e interactuar con las bases de datos. Interactúa con los datos para que sean almacenados y actualizados, a medida que avancemos, gestionandolos eficientemente y segura.

La elección de un sistema de gestión de bases de datos (DBMS) es muy importante, ya que depende de varios factores, como la complejidad de los datos, el rendimiento requerido, el presupuesto. Hay que tener en cuenta, que a la hora de nombrarlos se clasifican en diferentes categorías, según su estructura y el tipo de almacenamiento de datos utilizado. Algunos de los DBMS populares incluyen:

DBMS RELACIONALES:



- En **MySQL**: *DBMS* de código abierto basado en el modelo relacional. Es muy utilizado en aplicaciones web y es conocido por su velocidad y escalabilidad(sabremos que es despues).
- **Oracle Database**: Propiedad de Oracle Corporation, es un *DBMS* relacional de alto rendimiento y ampliamente utilizado en empresas. Combina elementos de los modelos de base de datos relacionales y orientados a objetos para proporcionar una mayor flexibilidad y capacidad de gestión de datos. NO es de código abierto. Ofrece una amplia gama de características y opciones.



- **PostgreSQL**: Como MySQL, es un *DBMS* de código abierto basado en el modelo relacional. Es conocido por su capacidad de gestión de datos geoespaciales y extensibilidad.



DBMS NoSQL:

- **MongoDB**: Un sistema de gestión de bases de datos NoSQL de código abierto que almacena datos en documentos BSON (Binary JSON). Es altamente escalable y adecuado para aplicaciones de big data.



- **Neo4j**: Un *DBMS* de grafos que se especializa en almacenar y consultar datos relacionales. NO es de código abierto. Es adecuado para aplicaciones de relaciones complejas.



DBMS DE MEMORIA:

- **SQLite**: Un *DBMS* ligero y de código abierto que se utiliza comúnmente en aplicaciones móviles y embebidas(insertión de contenido de otro origen en una página web o aplicación).



Los criterios de evaluación a la hora de elegir un DBMS incluyen:

- **Tipo de datos y modelo de datos requeridos:** Hay que saber qué tipo de datos vas a almacenar y cómo se relacionan entre sí. Ya que hay DBMS para datos tabulares y relacionales, mientras que otros son para datos no estructurados o grafos.
- **Escalabilidad:** Hay que considerar si el DBMS que hayas escogido puede escalar para manejar el crecimiento futuro de datos. Con esto nos referimos a la capacidad del DBMS para adaptarse y crecer a medida que se incrementa la cantidad de datos.
- **Rendimiento:** Hay que evaluar el rendimiento del DBMS en cuanto a la velocidad de lectura y escritura, a la capacidad de respuesta de las consultas y a la capacidad de procesamiento. Esto es primordial para garantizar que la aplicación funcione correctamente.
- **Características de seguridad:** La seguridad de los datos es fundamental. Hay que tener en consideración que el DBMS ofrezca características de seguridad sólidas, como autenticación, autorización, cifrado de datos y auditoría de registros. Ya que en aplicaciones que manejan datos es mejor que sean confidenciales o regulados.
- **Soporte de la comunidad o proveedor:** Revisa si el DBMS tiene una comunidad activa, si es de código abierto o un proveedor confiable que ofrezca soporte técnico, actualizaciones y parches de seguridad. La documentación en línea también es crucial, para saber las instrucciones, funciones y demás de la aplicación.
- **Costos de licencia y mantenimiento:** Obviamente, hay que ojear los costos asociados con el DBMS, incluyendo licencias, soporte y mantenimiento a largo plazo. Algunos DBMS de código abierto pueden ser más económicos, incluso gratis, en términos de licencia, pero es importante evaluar y comparar los costos totales de cada DBMS.
- **Integración con tecnologías existentes:** Verifica si el DBMS que tengas se integra fácilmente con otras herramientas que utilizas, como lenguajes de programación, servidores de aplicaciones y otras herramientas de análisis de datos.
- **Requisitos de almacenamiento y capacidad de gestión de datos:** Asegúrate de que el DBMS que utilices sea capaz de gestionar la cantidad de datos que quieras almacenar. No es lo mismo hacer una base de datos, con múltiples tablas y millones de datos, que una con pocas.

La elección de un DBMS eficiente debe basarse en estos factores para satisfacer tus necesidades y crear un proyecto, acorde a tu idea de un proyecto o aplicación.

2º PARTE:

L02 Diseñar un sistema de gestión de bases de datos utilizando un modelo relacional para satisfacer los requisitos del cliente.

Redacta tu propio escenario indicando los requisitos de usuario, es decir, decide de que va la aplicación que vas a desarrollar.

P2 Producir un diseño para un sistema de gestión de base de datos relacional para cumplir con los requisitos del cliente.

Diseña una base de datos (diseños conceptual, lógico y físico) que sirva para satisfacer los requisitos de usuario dentro del escenario que has planteado. Procura que no sea excesivamente compleja.

M2 Analizar cómo el diseño optimizará el rendimiento del sistema.

Planifica en la base de datos aquellas acciones que garanticen la seguridad y el rendimiento.

Explica en este apartado todas estas acciones. Ejemplos:

- ¿Cómo estás garantizando la integridad referencial?
- ¿Has planificado un plan de copias de seguridad?
- ¿Has creado distintas cuentas de usuario con distintos privilegios?
- etc.

LO2 Diseñar un sistema de gestión de bases de datos utilizando un modelo relacional para satisfacer los requisitos del cliente.

El escenario de nuestra aplicación: Sistema de Gestión de Inventario para Estancos

Yo soy un desarrollador encargado de crear un sistema de gestión para una cadena de estancos que opera en cuatro sucursales diferentes distribuidas por España. El sistema tiene como objetivo principal optimizar la gestión de productos, clientes, compras, empleados y tiendas. La empresa busca una solución eficiente para mejorar la experiencia del cliente, gestionar inventarios y supervisar el rendimiento del personal.

IBIZA	MADRID	ALICANTE	BARCELONA
			

A continuación se detallan los requisitos del usuario:

Requisitos del Usuario:

Clientes:

- Los clientes deben poder registrarse en el sistema.
- Acceder a su historial de compras.
- Actualizar su información personal.
- Recibir notificaciones sobre ofertas y promociones.

Empleados:

- Registrar nuevas compras realizadas por los clientes.
- Gestionar el inventario de productos en la tienda.
- Visualizar el desempeño de las ventas de la tienda.
- Acceder a la información de clientes para brindar un servicio personalizado.

Administradores:

- Supervisar el rendimiento general del sistema.
- Administrar permisos de usuario para empleados.
- Generar informes y estadísticas sobre las ventas y el inventario.
- Realizar consultas avanzadas para obtener información específica.

Funcionalidades Generales:

- Mantener la integridad referencial en todas las operaciones.
- Registrar registros de auditoría para cumplir con normativas como el GDPR.
- Realizar copias de seguridad periódicas y garantizar la recuperación de datos.
- Proporcionar una interfaz intuitiva para facilitar el uso del sistema.

Elementos Adicionales:

- La aplicación deberá permitir la asignación de productos a cada tienda.
- Se debe garantizar la consistencia de datos en todas las operaciones, evitando duplicados o inconsistencias.
- La seguridad de los datos es crucial, especialmente en lo que respecta a la información personal de los clientes.

Este escenario y sus requisitos proporcionan una base sólida para el diseño y desarrollo de nuestra base de datos, así como su modelo relacional que aborde las necesidades específicas del cliente en términos de gestión de inventario. Además de que la aplicación debe facilitar y satisfacer las necesidades específicas de la cadena de estancos.

P2 Producir un diseño para un sistema de gestión de base de datos relacional para cumplir con los requisitos del cliente.

Ahora viene la parte divertida. Crearemos una base de datos, con las necesidades de la cadena de estancos.

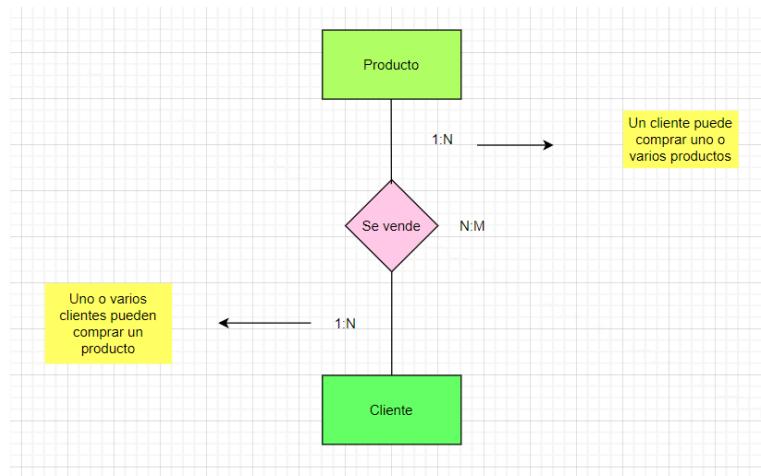
Primero crearemos un modelo conceptual de nuestra base de datos, describiendo de manera abstracta la estructura y las relaciones de los datos. Después estableceremos un modelo lógico del modelo anterior en Draw.io, representando el cómo se organizan los datos y cómo se establecerán las relaciones entre las tablas. Y por último, nos centraremos en el modelo físico, entrando más en detalle. En este caso, ya no utilizaremos una herramienta en línea, sino que será creado y probado en Oracle Database.

1º Modelo Conceptual:

Primero hay que tener claro que es un modelo conceptual y sus pasos para crearlo, ya que no podemos crearlo a ciegas, sin saber de qué se compone.

En el modelo conceptual de una base de datos se describen las principales actividades que se realizan en nuestra base de datos.

Primero tenemos que identificar nuestras **Entidades principales** del sistema. Estas entidades representan conceptos de la realidad. Basándonos en el ejemplo de la primera parte, tenemos la entidad *Producto* y la otra entidad *Cliente*.



También es necesario saber los **Atributos de las Entidades**. Estos atributos son características que describen las entidades. Deben ser claros en el diagrama, puesto en un óvalo. En la foto no aparecen, pero los atributos del Producto serían el nombre, su tamaño, su precio. Y del Cliente serían su nombre, su edad, su apellido, etc.

Otra parte importante sería las ***Relaciones entre Entidades***, ya que estas relaciones interactúan con las propias entidades, son las acciones por así decirlo y se deben escribir dentro de la relación. La relación debe ser expuesta con forma de rombo.

La Relación entre Producto y Cliente es se vende. Ya que el producto es vendido al cliente. En las relaciones es muy importante establecer su número. Pueden ser *de uno a uno(1:1)*, *uno a muchos(1:N)* o *muchos a muchos(N:M)*.

En el ejemplo establecido, se puede gestionar de la siguiente manera: "*Uno o varios productos se pueden vender a uno o varios clientes*"

Para poner uno o varios lo representamos como 1:N, el N representa varios, no se sabe cuántos. Esto para que luego en la relación se sumen los últimos números, es decir, se agarre la segunda parte (de 1:N, agarramos el N), de esta forma juntaremos las dos partes de la relación de las entidades, en la relación, de esta manera N:M(La forma de la M, es como si fuera una N, pero como ya hay una ponemos la M). Simbolizando que son *Uno o varios productos se pueden vender a uno o varios clientes*. Esto explicaremos más tarde para qué sirve. Por ahora solo explicaremos lo más básico.

Ahora debemos ***identificar las claves Primarias(PRIMARY KEY)***, estas claves están relacionadas con los atributos, ya que son atributos con más importancia que los demás, son únicos. Por ejemplo, de el Producto tenemos varios atributos, como hemos dicho antes tenemos el nombre, su tamaño y su precio, pero nada de esto nos sirve para identificar ese producto en concreto, ya que si por ejemplo tenemos varias sartenes, con el nombre de sartén no me sirve para reconocer una en concreto. Para reconocer una sartén es necesario tener su código de producto, que es diferente al resto de las demás sartenes.

Pues esto son las claves Primarias, de Producto podríamos escoger *codigo_producto*, como ya hemos dicho y de Cliente escogeríamos su *DNI*, ya que no habrá otro igual. Las claves primarias se establecieron igual que en los atributos pero deben estar subrayados.

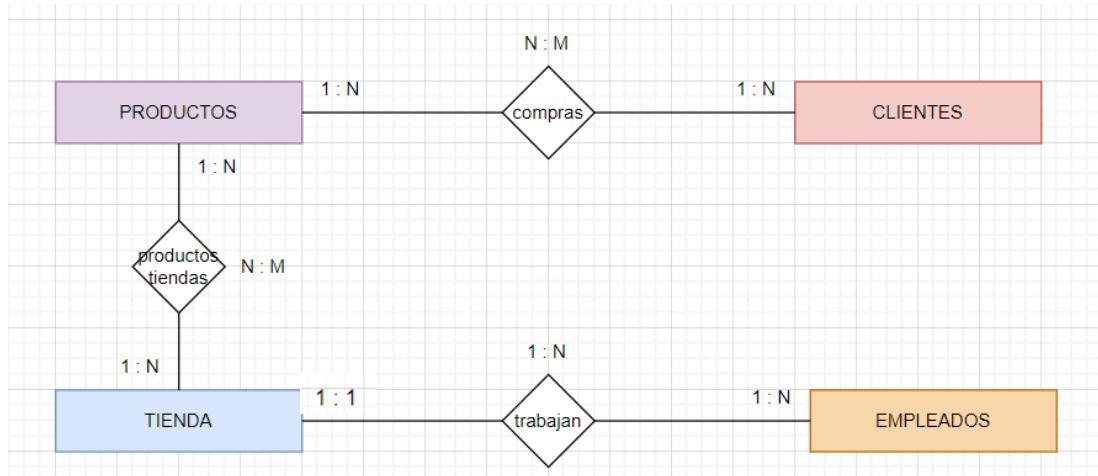
Hasta este punto, NO debemos crear ningún mapa conceptual, también llamado diagrama Entidad-relación todavía, sólo debemos apuntar los puntos anteriores. Ahora SÍ crearemos el diagrama.

Por último, **Revisaremos el diagrama** asegurandonos de que el modelo refleje con precisión las necesidades y requisitos de nuestro sistema.

En resumen, el modelo conceptual de una base de datos es una representación abstracta que establece la base para el diseño y desarrollo de la base de datos, brindando una comprensión clara de la estructura y las relaciones de los datos.

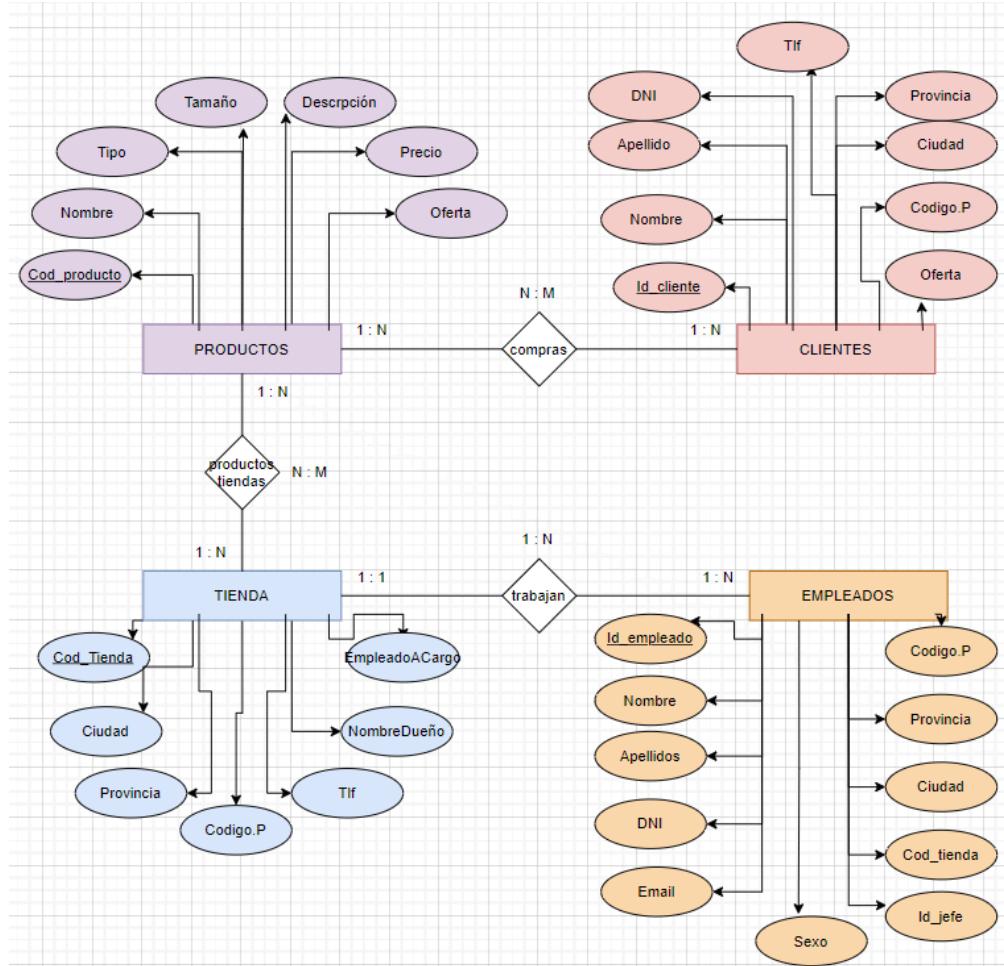
Ahora, en relación a nuestra base de datos, realizaremos nuestro diagrama Entidad-Relación con el modelo relacional.

Para especificar un poco más, nuestra base de datos será de cuatro locales, estancos, repartidos por España. En esos estancos, habrá unos productos que serán vendidos a los clientes. Y en esas tiendas habrá trabajando empleados. Básicamente es eso.



Este será nuestro modelo conceptual, sin los atributos para ver claramente el funcionamiento.

Ahora, si queremos ver los atributos, aquí está este otro diagrama donde se especifican mejor.



2º Modelo Lógico:

Ahora le toca al modelo lógico, o también llamado modelo Relacional, que como al modelo conceptual, lo definiremos lo mejor posible, expresando los pasos a realizar.

El modelo lógico de una base de datos se centra en la representación de la organización de los datos y de las relaciones entre las tablas en la base de datos, pero sin abordar en detalles. Aquí se describirán las principales actividades que se realizan en el modelo lógico de una base de datos:

Lo primero de todo que debemos hacer aquí y que no hemos hecho en el modelo conceptual es que además de saber las claves Primarias, debemos **saber las claves Foráneas(FOREIGN KEY)**. ¿Y qué son las claves Foráneas? Las claves foráneas serán los atributos que establecerán las relaciones entre tablas con sus claves primarias, actuando como vínculo.

Cuando una tabla tiene una clave foránea, significa que los valores en esa columna deben coincidir con los valores en la columna de la clave primaria de otra tabla.

Esto permite mantener la integridad de los datos y establecer conexiones entre la información almacenada de las diferentes tablas.

Si tenemos una tabla de Clientes con una clave primaria *ID_Cliente* y otra tabla de Productos con una clave primaria *ID_Producto*, puedes agregar una clave Foránea *ID_Cliente* en la tabla de *Productos*. Esto establece una relación entre los clientes y los productos, indicando qué cliente compró cada producto.

Ahora, después de saber nuestras claves foráneas, podemos **aplicar la Normalización**. Para garantizar la eficiencia y la integridad de la base de datos, organizando los datos y eliminando lo que no necesitemos. No vamos a hablar más de esto ya que lo vimos en la Primera Parte.

Después, ya podemos **crear las Tablas**. Estas tablas representarán las Entidades del modelo conceptual. Se crean básicamente primero como un cuadrado. Y aquí aplicaremos lo dicho en el modelo conceptual, exactamente en el apartado de las relaciones. A continuación colocaré unas notas importantes que se deben aplicar con las relaciones.

- R 1:1 => SE AÑADE LA CLAVE Y LA ENTIDAD A LA OTRA ENTIDAD (DA IGUAL A CUAL)
- R 1:N => CLAVE QUE MENOS PARTICIPA(0 o 1), SE AÑADE A LA ENTIDAD QUE MÁS(N)
- R N:M => SE CREA UNA ENTIDAD DE LA RELACIÓN + CLAVE Y ENTIDADES DE LAS MISMAS

Esto significa que en nuestro modelo entidad relación, como tenemos dos relaciones N:M, debemos crear dos entidades más, es decir, dos tablas más. Una llamada *Compras* y otra llamada *Productos_Tienda*. En las cuales tendrán como atributos, sus respectivas claves primarias con las que se hayan relacionado. En *Productos_Tienda* tendrá como atributos *cod_tienda* y *cod_producto*. Y en *Compras* tendrá *id_cliente* y *cod_producto*, pero también contará con otros atributos que le añadimos, como *Forma_pago*, *Fecha_pago*, *Cantidad* y *Oferta*.

Esas notas también significan que Tienda, exactamente *cod_Tienda* se añadirá a Empleados.

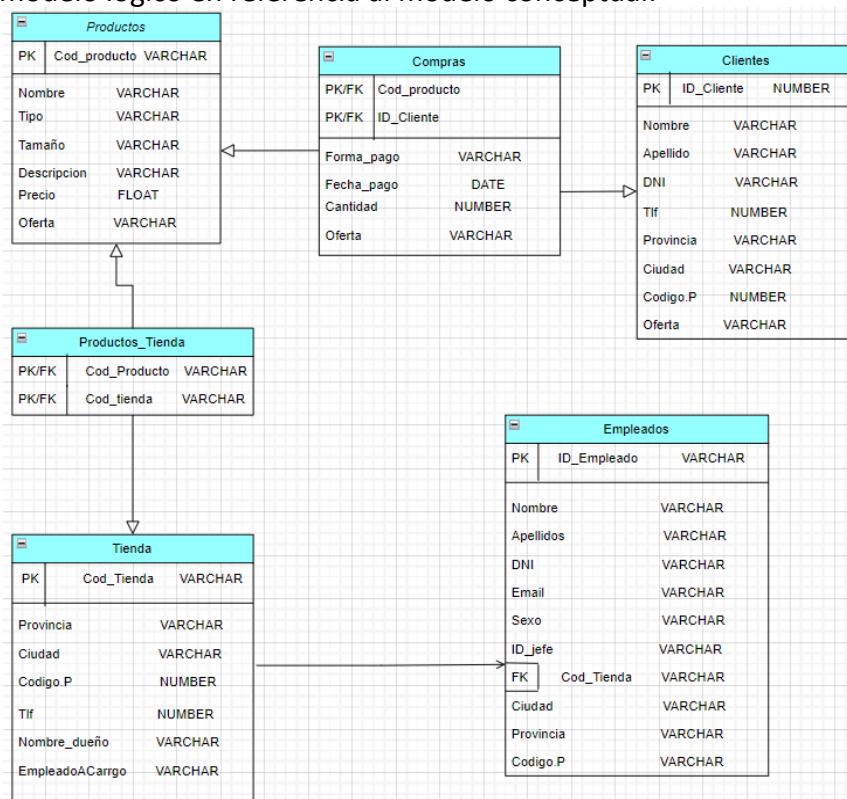
Luego dentro de las tablas **añadiremos los atributos**. Deberías añadir un índice, ordenar alfabéticamente o como tu veas oportuno los atributos, ya que esto es más opcional. Es muy importante, que los atributos vayan acompañados de su tipo de dato, ya sea INTEGER(número entero), VARCHAR(texto), DATE(fecha), etc. Esto sirve para detallar cómo se almacenarán los datos.

Posteriormente, **añadiremos las claves Primarias y Foráneas**. Normalmente se crea una columna al lado de los atributos y con poner PK o FK, Primary Key y Foreign Key, para saber cual es cual nos vale. Aunque nosotros nos ahorraremos esa columna.

Cabe destacar, que en este modelo no existen las relaciones con su determinado rombo y su frase. Aunque a muchas versiones, en una puede aparecer las frase, en otra el número de la tabla(1:N), o incluso ambas, pero lo que ya no hay son las formas geométricas. En este caso **asignaremos las relaciones entre las tablas** con líneas normales. En nuestro caso también añadiremos el número en relación a las tablas, para que se vea bien.

Por último y más importante, revisaremos el diagrama y nos aseguramos de que se cumplan los requisitos que implementamos.

Este es nuestro modelo lógico en referencia al modelo conceptual:



Como podemos observar, están las tablas prácticamente creadas. Con el añadido de *Compras* y *Productos_Tienda*. En ellas están sus claves primarias y foráneas de referencia. No solo en las nuevas tablas, si no también en Empleados, como ya hemos dicho anteriormente.

3º Modelo Físico:

Por último y más importante tocará el modelo físico.

El modelo físico convierte las decisiones tomadas en el modelo lógico en instrucciones específicas del sistema de gestión de bases de datos(DBMS) para implementar y optimizar la base de datos en un entorno de producción.

El modelo físico se ocupa de los detalles específicos de implementación, considerando aspectos como el rendimiento, la optimización de consultas y la asignación física de datos en el disco. Ya no es solo la creación de un diagrama. Aquí utilizaremos código de SQL para las creaciones de las tablas.

Todo esto lo veremos a continuación:

Lo primero de todo, es ***Crear una Cuenta de Usuario con una Contraseña*** que podamos recordar, pero que sea complicada, además de ir cambiandola cada cierto tiempo. Para hacer esto la propia aplicación que utilices te da la opción o tendrás que hacerlo manualmente

Una vez creada nuestra cuenta debemos saber ***Los Datos que Añadiremos***. En el modelo conceptual deberíamos tener los datos con los que rellenaremos los atributos, pero en este punto es mejor revisarlos.

En mi caso añadiré a continuación algunos de ellos:

Productos:

Mecheros (clipper, soplete, Zippo, Electrico)

Papelos normales (OCB, Raw, Juicy, Bambú, Randy's, Glass, Monkey king, trip)

Papelos de piti (Abadie, Rasta, OCB, RAW, Smoking, Greengo, Pay-Pay)

Tabaco de liar (Camel, Virginia, Horizon, Amberlif, Pueblo, Stanley, Sauvage, Chesterfield)

Tabaco industrial (Camel, Fortuna, West, Marlboro, Burton, Ducados, Mark, Winston, Chester, L&M)

Puros (Cohiba, diplomaticos, Montecristo, Punch Short de Punch, Partagás, Romeo y Julieta Petit Churchills, Partagás Legado, Partagás Culebras)

Sabor vaper (Fresa, Chocolate, Cannabis, Arándano, Tarta de queso, Limón, Sandía, Cerveza, Natilla, Vainilla, Plátano, Kiwi, Manzana, Mango)

Vaper (Bali fruits, Vaporesso, Shake It, Len & Jennys, Bud Vape, Hyppe Plus, Quawins y King Crest)

Cachimbas (Aladín → conocida por sus buenos materiales y precios asequibles.

Mr Shisha → una marca innovadora que ofrece productos exclusivos y de alta calidad a precios bajos.

Sky Hookah → una de las primeras marcas en fabricar sus modelos íntegramente con acero inoxidable V2A y en añadir un difusor regulable de 3 niveles al tiro.

Moze → conocida por sus materiales de alta calidad y su estética única.

Steamulation → se caracteriza por ofrecer cachimbas fabricadas en acero inoxidable V2A combinado con platino, lo que garantiza una larga durabilidad y evita problemas de corrosión.)

Refrescos (Fanta: Limón/naranja/sandia/piña/frambuesa, cocacola, Aquarius, red bull, Eneryeti, Monster, Zumo de piña/uva/melocotón, batido de chocolate)

Nueces (Natura)

Tarjetas de transporte.

Tarjetas prepago (Spotify, Play Store, Amazon, PlayStation,).

Libros (Mortadelo y Filemon, Don

No voy a añadir más aquí, pero puedes hacerte una idea.

Una vez tengamos los datos muy claros, debemos **Elegir el Espacio de Almacenamiento** en disco, el sistema de archivos, para las tablas. Obviamente teniendo una idea clara de los datos, tablas, etc. Aunque esto es opcional, ya que esto puede depender de la aplicación que utilicemos.

Ahora empezaremos a **Crear nuestra Base de Datos SQL** en la aplicación que a ti te parezca oportuna, dependiendo de tus necesidades.

Presentaremos el modelo que vamos a utilizar:

```
CREATE TABLE NombreTabla (
ID_primary  VARCHAR2(15) NOT NULL,
Nombre  VARCHAR2(10),
Numero  NUMBER(2,0),
Numero_decimal  FLOAT,
fecha  DATE,
tipo  VARCHAR2(10),
cod_atributo2  VARCHAR2,
PRIMARY KEY (ID_primary),
```

```
FOREIGN KEY (cod_atributo2 ) REFERENCES Tabla2 (cod_atributo2)
) ;
INSERT INTO nombreTabla VALUES ('ID','Nombre',numero, numero_decimal,
'fecha', NULL, 'VM');
```

Aquí hay mucho que explicar, así que iremos poco a poco. (No es necesario cambiar las letras a MAYÚSCULAS, no influye en nada, solo lo hice para que sea más visual y para organizar la estructura como es debido)

```
CREATE TABLE NombreTabla(
```

Crea la tabla con el nombre que deseas, para inicializarla con el paréntesis.

```
ID_primary VARCHAR2 (15) NOT NULL,
```

Creamos la tabla, empezando por las columnas.

Lo primero que debemos hacer es empezar con la que se considere o consideren claves primarias, en este caso es `ID_primary`. Con tipo de dato de tipo VARCHAR, en este caso es VARCHAR2 ya que es la versión actualizada del varchar normal. Como ya sabemos, VARCHAR2 es de tipo texto, como String. Lo que viene en el paréntesis de después es el número máximo de caracteres que debería tener. Es decir, que no puedes escribir una frase que supere esa cifra. Posteriormente, escribimos `NOT NULL`, Esto significa que NO puede ser un valor nulo, es decir, que tiene que haber escrito algo obligatoriamente.

Y MUY IMPORTANTE, cuando hayas terminado con esa columna poner una coma(,) al final para pasar a editar la siguiente columna.

Más abajo daremos a entender que esta es la clave primaria. Justo antes de poner la clave foránea.

```
PRIMARY KEY (ID_primary),
```

```
Nombre VARCHAR2 (10) ,
```

Otro atributo de tipo varchar2, llamado nombre, con 10 caracteres máximos. Y terminamos en coma. Finish, esto ya lo sabemos.

```
Numero NUMBER (2,0) ,
```

Ahora pasamos a otro tipo de dato, esta vez de tipo NUMBER, de tipo número entero o decimal. Lo que hay entre paréntesis es **1º (2)** ⇒ número máximo de dígitos **2º (0)** ⇒ cantidad máxima de dígitos en la parte decimal. Terminamos en coma.

NUMBER (precisión, escala)

En este NUMBER no habrá números decimales.

Numero_decimal **FLOAT**,

Ahora miraremos FLOAT, que guarda números decimales. ¿Y cuál es la diferencia con NUMBER? Pues qué NUMBER solo coje valores decimales con precisión, mientras que FLOAT lo hace más a lo loco, ya que si no sabes el número de decimales que necesitas, esta es la mejor opción.

fecha **DATE**,

Ahora añadiremos una fecha. Para añadir una fecha, esta debe ser construida con su determinada sintaxis: 'DD-MON-YYYY' → DD= días, MON = meses, YYYY = año. Pero esto lo veremos más adelante.

cod_atributo2 **VARCHAR2**,

Esto como vemos, puede ser perfectamente un atributo de otra tabla, siendo una clave foránea. Y como veremos un poco más adelante, es cierto.

FOREIGN KEY (cod_atributo2) **REFERENCES** Tabla2 (cod_atributo2)

Para estructurar una clave foránea se debe poner primero el nombre(**FOREIGN KEY**) y luego entre paréntesis poner a qué atributo te refieres. Después pones **REFERENCES** Tabla2, significando que estas haciendo referencia a la Tabla número 2 y cojiendo su atributo (cod_atributo2). Las claves foráneas suelen terminar este proceso de generar columnas, asi que aqui ya no terminamos con coma.

Además si quieres añadirle un nombre a esta clave Foránea, antes de poner el nombre de la restricción(la clave foránea pertenece a las restricciones) añades **CONSTRAINT** y un nombre que lo represente, por ejemplo:

CONSTRAINT claveF **FOREIGN KEY** (cod_atributo2) **REFERENCES** Tabla2 (cod_atributo2)

) ;

Por último cerramos la edición de las columnas de este modo y nos centramos en añadir los datos.

INSERT INTO nombreTabla **VALUES** ('ID', 'Nombre', numero, numero_decimal, 'fecha', NULL, 'VM');

Empezamos por **INSERT INTO** nombreTabla **VALUES**, Esto nos da a entender lo siguiente: "Insertamos dentro de nombreTabla estos valores()"

1. Para añadir valores en VARCHAR, es decir, en texto, debemos añadir comillas simples (''), sea lo que sea.
2. LOS SEPARAMOS EN COMAS
3. Los números enteros o decimales se añaden solos.
4. Para la fecha: Como ya hemos dicho, debemos añadirla con comillas simples y su determinada estructura, 'DD-MON-YYYY', por ejemplo: '23-1-2023'

tipo VARCHAR2 (10),

5. NULL. ¿Qué significa si en los valores encontramos un NULL? Se traduce que es un valor nulo, es decir, que se queda en blanco. No hay valor para él. Si hubiera en tipo un NOT NULL, como en la clave primaria esto no se podría hacer.
6. Por último, la clave foránea, como la clave primaria se añaden como valores normales. Pero poniendo el mismo valor que en su determinada tabla, en este caso Tabla2.

Este será nuestro modelo de base de datos.

A continuación adjunto un ejemplo:

```
CREATE TABLE JugadoresFutbol (
DNI  VARCHAR2 (9) NOT NULL,
Nombre VARCHAR2 (20),
Mayor_valocidad NUMBER (3, 2),
fecha_nacimiento DATE,
Nacionalidad VARCHAR2 (20),
Numero NUMBER (2, 0),
PRIMARY KEY (DNI)
);
INSERT INTO JugadoresFutbol VALUES ('40234198M', 'Sergio', 40.33,
'23-09-1970', 'Español', 20);
INSERT INTO JugadoresFutbol VALUES (...)
```

Una vez creada la base de datos **Aplicaremos Procesos de Seguridad Apropriados**. Cómo guardar una copia de seguridad almacenando los datos u optimizar el rendimiento.

Y por último, como ya hemos visto toca la **Revisión del Código**, asegurándonos de que está todo correcto y **Validación del usuario** añadido.

Una vez entendido todo, toca presentar nuestra base de datos propia. Para eso, adjunte un bloc de notas llamado *Tienda_Estanco_Definitivo*. Ya que la base de datos tiene mucho contenido y si lo añadiese aquí sería un documento demasiado largo.

En ese archivo encontraremos una base de datos de un estanco, con seis tablas, algunas unidas con claves foráneas y otras sin necesitarlo.

Ahora la probaremos para ver si realmente ha salido todo bien.

M2 Analizar cómo el diseño optimizará el rendimiento del sistema.

Terminando ya esta última parte, hay que dejar claro que la planificación en una base de datos desempeña un papel crucial al asegurar la seguridad y optimizar el rendimiento del sistema. Este proceso estratégico abarca diversas acciones que se centran en mantener la integridad de los datos, prevenir posibles problemas de rendimiento y garantizar un entorno operativo seguro. Aquí se exploran algunos aspectos clave:

Cuentas de Usuario y Privilegios:

La creación de cuentas de usuario y contraseñas varían según el sistema de gestión de bases de datos (SGBD) que estés utilizando. Utiliza contraseñas seguras y cambia las contraseñas de manera regular y limita el acceso a las tablas y funciones sensibles a cuentas específicas.. Algunos ejemplos son:

MySQL / MariaDB: ⇒ Crear una cuenta de usuario en MySQL o MariaDB, puedes usar el siguiente comando SQL:

```
CREATE USER 'nombre_usuario'@'host' IDENTIFIED BY 'tu_contraseña';
```

- `nombre_usuario`: El nombre de tu usuario.
- `host`: La dirección IP o nombre de host desde la cual el usuario puede conectarse (puedes cambiarlo a '%' para permitir conexiones desde cualquier lugar).
- `tu_contraseña`: La contraseña para el nuevo usuario.

```
CREATE USER 'marcosJ'@'localhost' IDENTIFIED BY 'MaRcOsJJ';
```

PostgreSQL: ⇒ Creamos un nuevo rol, parecido a un usuario:

```
CREATE USER 'nombre_usuario' WITH PASSWORD 'tu_contraseña';
```

Ejemplo:

```
CREATE USER 'marcosJ' WITH PASSWORD 'MaRcOsJJ';
```

SQL Server: ⇒ En SQL Server, utilizamos el siguiente comando para crear un nuevo usuario:

```
CREATE LOGIN nombre_usuario WITH PASSWORD = 'tu_contraseña';
```

Ejemplo:

```
CREATE LOGIN 'marcosJ' WITH PASSWORD = 'MaRcOsJJ'
```

Oracle: ⇒ En Oracle, puedes crear un nuevo usuario con un comando muy parecido a MySQL:

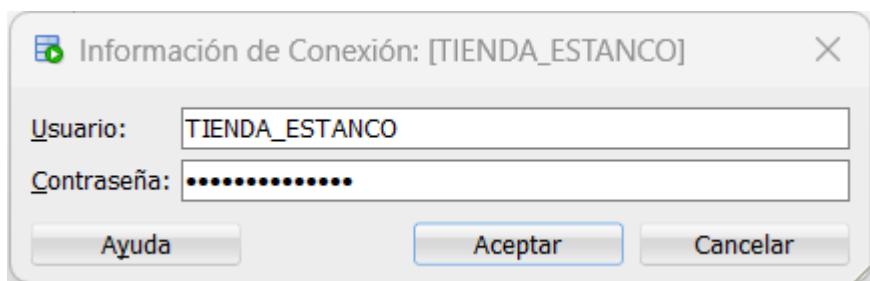
```
CREATE USER nombre_usuario IDENTIFIED BY tu_contraseña;
```

Ejemplo:

```
CREATE USER 'marcosJ' IDENTIFIED BY 'MaRcOsJJ';
```

Es importante que después de crear un usuario, generalmente también necesitarás otorgarle los privilegios necesarios, como *SELECT, INSERT, UPDATE, DELETE*, etc., dependiendo de los permisos que deseas otorgar al usuario en particular.

En nuestro caso con Oracle SQL Developer se utilizan herramientas externas para poner el usuario y la contraseña:



Para este caso puse de nombre de usuario el nombre de la base de datos y la contraseña igual. Pero esto no se debe hacer si de verdad quieres tener seguros tus datos.

Copias de Seguridad:

Planificamos un plan regular de copias de seguridad para proteger los datos en caso de pérdida o corrupción. Podemos utilizar herramientas de la propia aplicación de la base de datos, en código Cada aplicación tiene sus propios comandos para realizar este objetivo. O también se pueden utilizar herramientas externas de la aplicación para realizar copias de seguridad.

Cada aplicación es distinta y suelen tener distintos comandos para realizar una copia de seguridad:

MySQL → ***mysqldump***: Esta herramienta interna de MySQL realiza copias de seguridad y restauraciones.

- **Copia de Seguridad:**

```
mysqldump -u usuario -p nombre_bbdd > backup.sql
```

- **Restaurar:**

```
mysql -u usuario -p nombre_bbdd < backup.sql
```

PostgreSQL → ***pg_dump***: Es una herramienta de línea de comandos integrada en PostgreSQL para realizar copias de seguridad y restauraciones.

- **Copia de Seguridad:**

```
pg_dump -U usuario -h localhost -d nombre_bbdd -F c -f backup.dump
```

- **Restaurar:**

```
pg_restore -U usuario -h localhost -d nombre_bbdd -F c -c -v backup.dump
```

SQL Server → ***Backup y Restore***: SQL Server tiene comandos nativos como **BACKUP DATABASE** y **RESTORE DATABASE** que puedes ejecutar para realizar copias de seguridad y restauraciones.

- **Copia de Seguridad:**

```
BACKUP DATABASE nombre_bbdd TO DISK = 'C:\Ruta\backup.bak';
```

- **Restaurar:**

```
RESTORE DATABASE nombre_bbdd FROM DISK = 'C:\Ruta\backup.bak';
```

Oracle → ***Data Pump***: Es una herramienta interna de Oracle para la exportación e importación de datos.

- **Copia de Seguridad:**

```
expdp usuario/contraseña@nombre_bbdd DIRECTORY= directorio_copia_de_seguridad DUMPFILE=backup.dmp SCHEMAS=esquema_a_copiar
```

- **Restaurar:**

```
impdp usuario/contraseña@nombre_bbdd DIRECTORY= directorio_copia_de_seguridad DUMPFILE=backup.dmp SCHEMAS=esquema_a_restaurar
```

MongoDB → ***mongodump*** y ***mongorestore***: Ambas son herramientas de línea de comandos que vienen con MongoDB y se utilizan para realizar copias de seguridad y restauraciones.

- **Copia de Seguridad:**

```
mongodump --db nombre_bbdd --out directorio_copia_de_seguridad
```

- **Restaurar:**

```
mongorestore --db nombre_bbdd directorio_copia_de_seguridad /nombre_bbdd
```

En nuestros caso utilizaremos estos comandos para Oracle SQL developer para hacer una copia de seguridad.:

1º Abre una ventana de comandos o terminal.

2º Inicia *RMAN* ejecutando el siguiente comando y proporcionando las credenciales adecuadas:

```
rman target username/password@database
```

3º Sustituye *username*, *password*, y *database* con tus propios valores.

4º Ejecuta el siguiente comando dentro de RMAN para realizar una copia de seguridad completa:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

Este comando realiza una copia de seguridad de la base de datos y los archivos de registro (archive logs).

5º Puedes finalizar RMAN con el siguiente comando: `EXIT;`

También podemos utilizar para realizar una base de datos:

```
BACKUP DATABASE HITOEstanco TO DISK =  
'C:\Users\marqu\OneDrive\Documentos\BBDD\HITOEstanco.sql ';
```

Para restaurar la base de datos ejecutamos este comando:

1º Realiza los pasos anteriores hasta el paso **3º**

2º:

Si estás recuperando la base de datos a un punto específico en el tiempo, utiliza el siguiente comando en RMAN:

```
RUN {  
    SET UNTIL TIME 'fecha_y_hora';  
    RESTORE DATABASE;  
    RECOVER DATABASE;
```

- Asegúrate de reemplazar '`fecha_y_hora`' con la fecha y hora hasta la cual deseas realizar la recuperación.

Si estás recuperando hasta el último backup(última copia de seguridad), simplemente puedes ejecutar:

```
RESTORE DATABASE;  
RECOVER DATABASE;
```

3º Abrir la base de datos con este comando:

```
ALTER DATABASE OPEN RESETLOGS;
```

Esto abrirá la base de datos con un nuevo conjunto de registros de redo.

Salir de RMAN:

Puedes salir de RMAN cuando hayas completado el proceso de recuperación:

```
EXIT;
```

Optimización de Rendimiento:

1. Realizamos un análisis del rendimiento de las consultas para identificar y optimizar aquellas que puedan afectar el rendimiento.

- Utilizar herramientas de monitoreo y perfiles:

```
SELECT * FROM tabla WHERE condicion;
```

- Examinar los planes de ejecución:

```
SHOW FULL PROCESSLIST;
```

- Considerar el uso de herramientas de análisis de rendimiento como *pt-query-digest* para examinar consultas lentas.

2. Indexa adecuadamente las columnas que se utilizan con frecuencia en operaciones de búsqueda.

- Crear índices en columnas utilizadas en cláusulas *WHERE* y *JOIN*:

```
CREATE INDEX idx_nombre_columna ON tabla (nom_columna);
```

- Evitar el uso excesivo de índices:

Solo crea índices necesarios para mejorar el rendimiento y evita índices innecesarios que puedan afectar el rendimiento de las operaciones de escritura.

3. Considera la posibilidad de utilizar particiones si las tablas son muy grandes.

Las particiones es la división lógica o física de una tabla en subconjuntos más pequeños. Estos subconjuntos tienen sus propias características de almacenamiento y son tratados de manera independiente para mejorar el rendimiento de ciertas operaciones. Esta técnica de diseño avanzada, mejora la eficiencia en la gestión de grandes cantidades de datos.

- Crear una tabla particionada por rango:

```
CREATE TABLE tu_tabla_particionada (
    id INT,
    fecha DATE
) PARTITION BY RANGE (YEAR(fecha)) (
    PARTITION p0 VALUES LESS THAN (2000),
    PARTITION p1 VALUES LESS THAN (2001),
    PARTITION p2 VALUES LESS THAN (2002),
    ...
);
```

- Elegir una columna de partición que sea comúnmente utilizada en consultas.

Auditoría:

Implementa la auditoría para rastrear actividades, como acceso a la base de datos, cambios en los datos, etc. Nos centraremos en Oracle.

1. Habilitar la Auditoría:

- Habilitamos la auditoría sólo para sesiones:

```
ALTER SYSTEM SET AUDIT_TRAIL=DB SCOPE=SPFILE;
```

2. Reiniciar la Base de Datos:

- Reiniciamos la base de datos para aplicar cambios

```
SHUTDOWN IMMEDIATE;  
STARTUP;
```

3. Configurar Auditoría para Acceso:

- Auditoría de intentos de inicio de sesión (incluyendo fallidos)
AUDIT CREATE SESSION;

- Auditoría de cambios de roles

```
AUDIT BY ACCESS, ROLE;
```

- Auditoría de cambios de privilegios

```
AUDIT BY ACCESS, PRIVILEGE;
```

4. Configurar Auditoría para Cambios en Datos:

- Auditoría de cambios en una tabla específica

```
AUDIT SELECT, INSERT, UPDATE, DELETE ON hr.employees;
```

- Auditoría de todos los cambios en la base de datos

```
AUDIT ALL ON hr.schema;
```

- Para detener la auditoría en una tabla específica

```
NOAUDIT SELECT, INSERT, UPDATE, DELETE ON hr.employees;
```

5. Consultar Información de Auditoría:

- Consultar los registros de auditoría

```
SELECT * FROM dba_audit_trail;
```

- Consultar auditoría específica para un usuario

```
SELECT * FROM dba_audit_trail WHERE username =  
'nombre_usuario';
```

Todo esto ya no lo haríamos en la propia base de datos, sino que se haría en otra hoja fuera de la misma.

3º PARTE:

LO3 Desarrollar un sistema de gestión de bases de datos utilizando una plataforma adecuada.

P3 Desarrollar un sistema completamente funcional que cumpla con los requisitos del cliente y del sistema, utilizando un lenguaje de código abierto con un software de aplicación.
Aquí es donde debéis desarrollar vuestra aplicación Java. La desarrollaréis utilizando la librería JDBC para el acceso a la base de datos. Cómo interfaz de usuario, podéis crear un programa de consola, igual que en el hito de la cuenta bancaria, aunque sí alguien se atreve, también puede hacer una aplicación de escritorio (no obligatorio).

P4 Pruebe la funcionalidad y el rendimiento del sistema.

Realiza una batería de pruebas para buscar posibles errores, documenta esta batería de pruebas. Después corrige los errores localizados. Documenta el proceso.

M3 Implementar funciones eficaces en la solución para gestionar la concurrencia, la seguridad, las autorizaciones de usuarios y la recuperación de datos.

Pon en marcha funciones dentro de tu programa relacionadas con la seguridad y la confidencialidad. Debes reflejar estas funciones en la memoria de tu proyecto.

D2 Evaluar la eficacia del diseño y desarrollo del sistema con respecto a los requisitos del cliente y del sistema.

Realiza de nuevo una batería de pruebas, pero esta vez cada prueba irá asociada a un requisito de usuario. Se debe documentar cómo se ha comprobado que se cumple cada requisito de usuario.

Ahora nos toca añadir nuestra base de datos creada a una aplicación que utilice Java, para poder crear nuestra propia aplicación.

LO3 Desarrollar un sistema de gestión de bases de datos utilizando una plataforma adecuada.

Para realizar nuestra aplicación utilizaremos Eclipse. Básicamente porque es muy cómoda para mi y porque es la que hemos estado utilizando.

Pero, ¿Qué es Eclipse?

Eclipse es un entorno de desarrollo integrado (IDE) muy utilizado para el desarrollo de software en Java y otras tecnologías. Aquí hay información sobre Eclipse y su uso en el contexto de Java:

Eclipse es conocido por ser un IDE personalizado, con el que puedes ampliar sus funciones mediante la instalación de complementos (plugins). Además de ser Multiplataforma, ya que es compatible con varias plataformas, permitiendo trabajar en sistemas operativos como Windows, macOS y Linux.

Eclipse es ampliamente utilizado para el desarrollo de proyectos Java, desde aplicaciones básicas hasta proyectos empresariales complejos. Proporciona un potente editor de código Java para funciones, sintaxis, autocompletado y navegación fácil. Además, facilita la gestión de proyectos Java, con herramientas para organizar archivos, bibliotecas y dependencias.

Bueno, en resumen, Eclipse es una opción popular para el desarrollo de software Java debido a su flexibilidad, extensibilidad y conjunto de características robustas. Muchos desarrolladores encuentran Eclipse especialmente útil para proyectos Java de diversos tamaños y complejidades.

Para aplicar nuestra base de datos creada anteriormente a Eclipse, usaremos un Driver JDBC.

El Driver son determinadas clases que permiten a tu aplicación comunicarse con una base de datos. EJ = [com.mysql.cj.jdbc.Driver](#) → controlador para MySQL. Al cargar este controlador, le estás diciendo a Java que estás preparado para conectarte a una base de datos, en este caso, MySQL.

Este Driver depende de la aplicación de bases de datos que estés utilizando.

En mi caso utilizo *ojdbc11.jar*, ya que yo utilizo Oracle SQL.

Los Drivers se consiguen si te los descargas desde la Pagina principal, ya sea de Oracle, MySQL u otro:

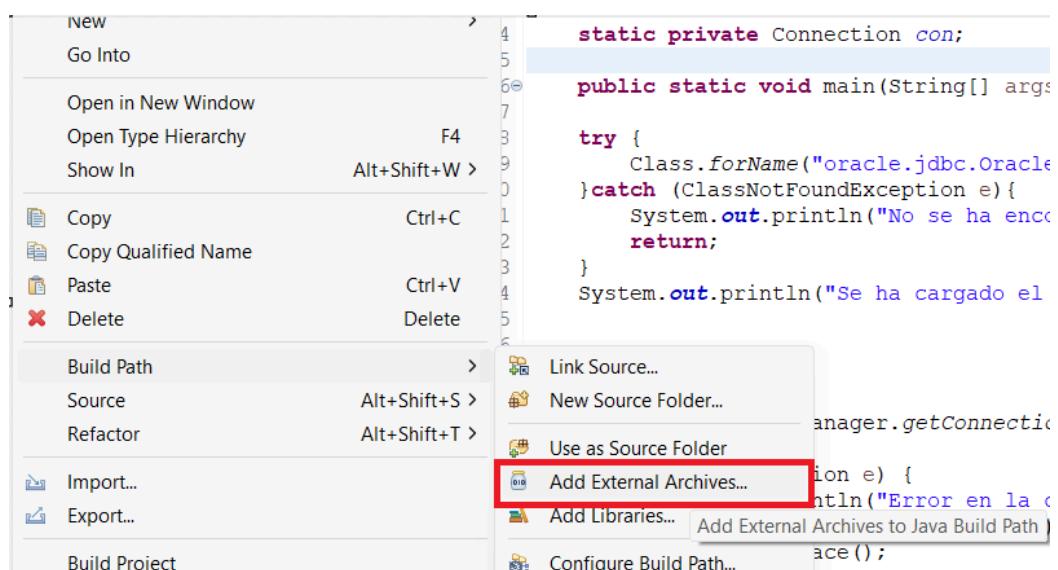
Oracle Database 23c (23.3.0.23.09) JDBC Driver & UCP Downloads - Versión gratuita

Es compatible con las versiones de Oracle Database: 23c, 21c, 19c, 18c y 12.2. Consulte JDBC-UCP-ReleaseNotes-23c.txt y [Bugs-fixed-in-23c.txt](#) Nota: Los controladores Oracle JDBC 23.3 sólo se admiten para uso de producción con BaseDB.

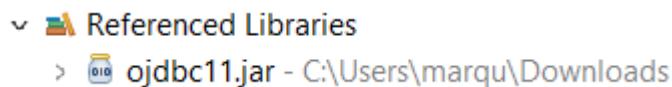
Los controladores de la versión 23.3 no son compatibles con la producción de ninguna otra configuración de base de datos.

Nombre	Descargar	Compatible con JDK	Descripción
Controlador JDBC de Oracle	ojdbc11.jar	Implementa la especificación JDBC 4.3 y está certificado con JDK11, JDK17, JDK19 y JDK21	Controlador JDBC de Oracle, excepto las clases para el soporte de NLS en los tipos de objeto y colección de Oracle. (7107784 bytes) - (SHA1: 405bcbc8d8dab59f562125fa1d2b7e06d21649f3)

Ese es el que yo me descargue para hacer el Trabajo. Despues de haberlo descargado, iremos a nuestro Proyecto Java y daremos click derecho sobre el. Pincharemos en Build Path> (la que tiene la flechita) y daremos a AddExternalJARs.



Después se nos abrirá Archivos, donde escogeremos nuestro Driver descargado y le pinchamos. Así se añadirá a nuestro proyecto.



P3 Desarrollar un sistema completamente funcional que cumpla con los requisitos del cliente y del sistema, utilizando un lenguaje de código abierto con un software de aplicación.

1º ACCESO A LA BASE DE DATOS

Para acceder a nuestra base de datos, tenemos que añadir varias cosas a nuestro código.

```
static private Connection con;

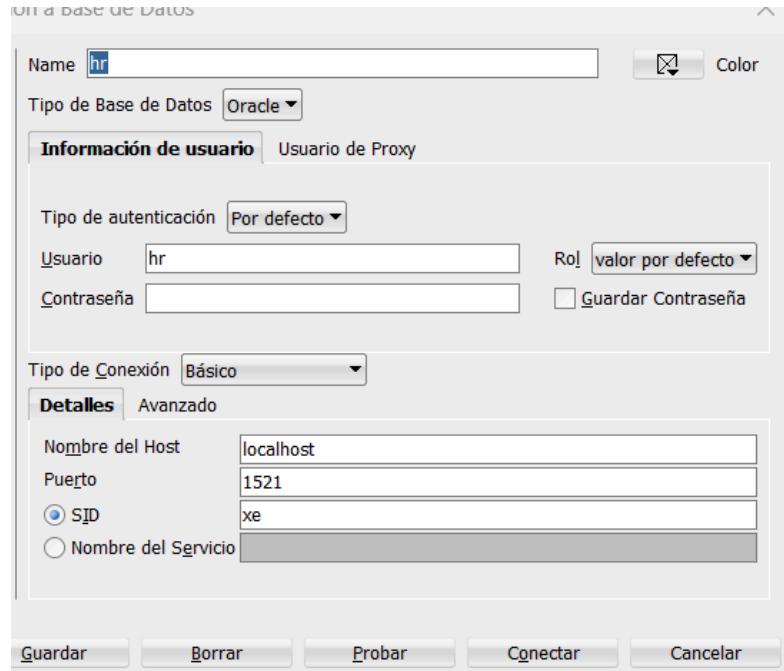
public static void main(String[] args) {
    try {
        Class.forName("oracle.jdbc.OracleDriver");
    } catch (ClassNotFoundException e) {
        System.out.println("No se ha encontrado el driver para MySQL");
        return;
    }
    System.out.println("Se ha cargado el Driver de MySQL");

    try {
        con =
        DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "hr",
        "hr");
    } catch (SQLException e) {
        System.out.println("Error en la conexión con la BD");
        System.out.println(e.getMessage());
        e.printStackTrace();
        return;
    }
    System.out.println("Se ha establecido la conexión con la Base de
datos");
}
```

Primero debemos saber cómo especificar el Driver en el código. Debemos importarlo, de manera que quede plasmado en el código y cargarlo en memoria con el método `Class.forName()`. Mi Driver es `oracle.jdbc.OracleDriver`. Una vez hecho eso y haber añadido las Excepciones necesarias, crearemos el método para conectarnos a la bbdd. Creamos `Connection`, que es con el que se conectara. Añadimos el `DriverManager`, que es la clase que administra los controladores de bbdd, junto con `getConnection`, el método que establecerá una conexión con la bbdd. Para este método se deben añadir tres elementos.

- cadena de conexión con detalles sobre la ubicación. Estos detalles son añadir `jdbc:` siempre al principio. Y después el tipo de aplicación que usaste para la bbdd, como ya he dicho, en mi caso yo he utilizado Oracle. Luego añadimos `:thin:` `@localhost`, o Esto porque estoy en mi mismo ordenador, si no pondría otra IP.

Posteriormente añadimos el puerto, sea cual sea y por último el SID, Identificador Del Sistema. Es un nombre único que identifica una instancia de una base de datos en un servidor. El mío es xe, sencillo. Si no tienes mucha idea de esto, no te preocunes. Si vas a tu bbdd, en mi caso si voy a Oracle y pincho sobre la instancia donde se encuentra mi bbdd y le doy a *Propiedades*, me dará esta información:



Tanto el nombre de usuario, como tu Host, además de tu puerto y SID.

- Luego habría que añadir tu nombre de usuario
- Y por último la contraseña. Yo he elegido un usuario y contraseña sencillos para facilitarme las cosas.

2º JUGAR CON LA BASE DE DATOS

Ahora ya podemos editar nuestra base de datos, extrayendo datos de ella. Como por ejemplo hago aquí:

```
public List<String> obtenerNombresDeProductos() throws
ClassNotFoundException {
    List<String> nombres = new ArrayList<>();
    try { //-----NOMBRE-----
        con = getConexion();
        String query = "SELECT Nombre FROM Productss";
        try (Statement st = con.createStatement();
             ResultSet rs = st.executeQuery(query)) {
            while (rs.next()) {
                nombres.add(rs.getString("Nombre"));
            }
        }
    } catch (SQLException e) {
        System.out.println("Error al realizar el listado de
nombres");
```

```

        System.out.println(e.getMessage());
    }
    return nombres;
}

```

Aquí lo que hago es primero, crear una Lista donde añada lo que voy a sacar de la bbdd. Obviamente necesito estar conectada a ella, por eso el `getConexion()`, que básicamente es un método que contiene todo lo explicado anteriormente.

Ahora jugamos con el `query`, en esta parte hemos añadido un `SELECT` para coger todos los nombres de los productos de nuestra bbdd.

En la parte de dentro del segundo `try{ Creamos un objeto Statement para ejecutar consultas SQL y otro objeto ResultSet para obtener los resultados de la consulta.`

Después creamos un bucle `while` para traer todos los nombres de la tabla, es decir, va línea por línea cojiendo todo los resultados con el `ResultSet` y cuando termina se cierra automáticamente.

Después de obtener esos resultados los añadimos a la lista de nombres.

A parte de la base de datos, yo he querido añadir Interfaces. Para que sea más visual. Lamentablemente no he podido completarlo al 100% de mis ideas, pero no ha quedado nada mal tampoco.

Por otra parte he decidido añadir un poco de hilos.

```

public class Inicio {
    public static void main(String[] args) {
        Accesobbdd acceso = new Accesobbdd();
        java.awt.EventQueue.invokeLater (new Runnable() {
            ProductosEstanco almacen = new ProductosEstanco();

            public void run() {
                VPEstanco vpes = new VPEstanco();
                VIIntroducir vi = null;
                try {
                    vi = new VIIntroducir(acceso, almacen);
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
                VConsultar vc = new VConsultar(almacen);
                EstancoListener listener = new EstancoListener(vpes, vi, vc,
                        almacen);

                vpes.setListener(listener);
                vi.setListener(listener);
                vc.setListener(listener);

                vpes.hacerVisible();

                java.awt.EventQueue.invokeLater(new Runnable() {

```

Aquí estamos utilizando el *Event Dispatch Thread (EDT)*, práctica común en las aplicaciones de interfaz gráfica de usuario (GUI) en Java. La interfaz de usuario de Swing debe manipularse en el hilo para evitar problemas de concurrencia.

Este hilo especial se usa en las aplicaciones de interfaz gráfica de usuario (GUI) de Java, sobre todo en el entorno de desarrollo Swing. Su objetivo principal es manejar eventos relacionados con la interfaz de usuario, como clics de ratón, pulsaciones de teclas y actualizaciones de componentes de la interfaz de usuario.

```
public void run() {
```

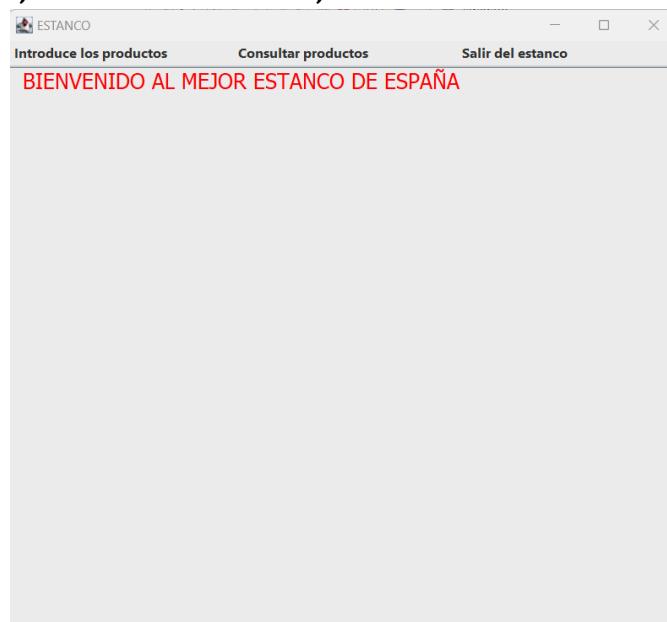
Dentro de la implementación del método `run` de la interfaz `Runnable`, se realizan varias operaciones.

```
} ); → Cierre del bloque del Event Dispatch Thread (EDT).
```

P4 Pruebe la funcionalidad y el rendimiento del sistema.

Bien, ahora toca la parte divertida. Donde probaremos si nuestro código funciona o no.

Al darle al PLAY o RUN, para que la aplicación empiece a funcionar nos presentara una primera parte, la interfaz de inicio, dándonos la bienvenida:



Como podemos ver arriba de la página, tenemos tres subMenús. *Introducir productos - Consultar Productos - Salir del estanco.*

Si pinchamos en el primero, es decir, en insertar productos nos aparecerá otra interfaz.

 ESTANCO

Introduce los productos Consultar productos Salir del estanco

BIENVENIDO AL MEJOR ESTANCO DE ESPAÑA

Tipo de producto:

Nombre del producto:

Descripción:

Cantidad:

Precio: €

Oferta:

COMPRAR

En esta parte viene lo bueno.

Como podemos ver arriba hay dos listas. En la primera, es donde están los tipos de productos y en la segunda sus nombres. Estos datos vienen directamente de la base de datos.

Podemos seleccionar los productos que queramos. Lo malo es que no he sabido estructurarlo de manera que si pulsas un tipo solo te salieran los de ese tipo, una lastima. Aún así también se puede decir que lo he hecho a posta ya que si no sabes de fumar mejor que no fumes. Al final lo hago por el bien de los demás. Aunque no solo hay tabaco. También hay libros, comida, bebida, tarjetas de prepago, para llenar el abono de transporte, etc.

Tipo de producto:	<input type="text" value="Sabor vaper"/>
Nombre del producto:	<input type="text" value="Cachimbas"/>
Descripción:	<input type="text" value="Refrescos
Nueces
Tabaco de liar
Puros
Targeta transporte
Libros
Papelazos"/>

Nombre del producto:	<input type="text" value="Fortuna"/>
Descripción:	<input type="text" value="Reo bull
Energyti
Monster
Zumo de piña
Zumo de uva
Zumo de melocotón
Batido de chocolate
Nueces
Targeta transporte"/>

Aquí viene la magia.

Cuando ya tenemos un tipo de producto seleccionado y un nombre de ese tipo aparecerán las demás opciones llenas. Como la Descripción, el Precio y si tiene ofertas o no. Pondré varios ejemplos:

Tipo de producto:	<input type="text" value="Refrescos"/>
Nombre del producto:	<input type="text" value="Fanta de sandia"/>
Descripción:	<input type="text" value="Energía: 7 kilocalorías. Grasas: 0 gramos. Hidratos de carbono: 1 gramo. Azúcares: 1 gramo. Proteínas: 0 gramos. Sál: 0,07 gramos."/>
Cantidad:	<input type="text" value="0"/>
Precio:	<input type="text" value="1.5 €"/>
Oferta:	<input type="text" value=""/>
COMPRAR	

Tipo de producto: Puros

Nombre del producto: Diplomaticos

Descripción:
Los puros se enrollan totalmente a mano en la fábrica de José Martí, al igual que algunos puros de Montecristo . Además, todos los puros de su catálogo son de capa larga. Las hojas de tabaco proceden exclusivamente de las plantaciones de Vuelta Abajo, tanto si se utilizan para la tripa como para la envoltura.

Cantidad: 0

Precio: 20 €

Oferta:

COMPRAR

Tipo de producto: Tabaco industrial

Nombre del producto: Fortuna

Descripción:
Tabaco de cigarrillos rubios con no muy buen sabor. Con 20, 21, 37 o 40 cigarros por caja

Cantidad: 0

Precio: 5 €

Oferta: 10 x 40.00 €

COMPRAR

Además de eso puedes añadirle la cantidad que tu prefieras. Pero de 1 en 1 (lo acabó de mirar) :

Tipo de producto: Sabor vaper

Nombre del producto: Fortuna

Descripción:

Cantidad: 1

Precio: €

Oferta:

Una vez tengamos todo colocado le daremos al botón de comprar. Si todo se ha guardado bien, saltará un mensaje comunicándolo, sí no, saltará uno de error.

Tipo de producto: Sabor vaper

Nombre del producto: Fortuna

Descripción:

Productos

El producto se ha guardado con éxito

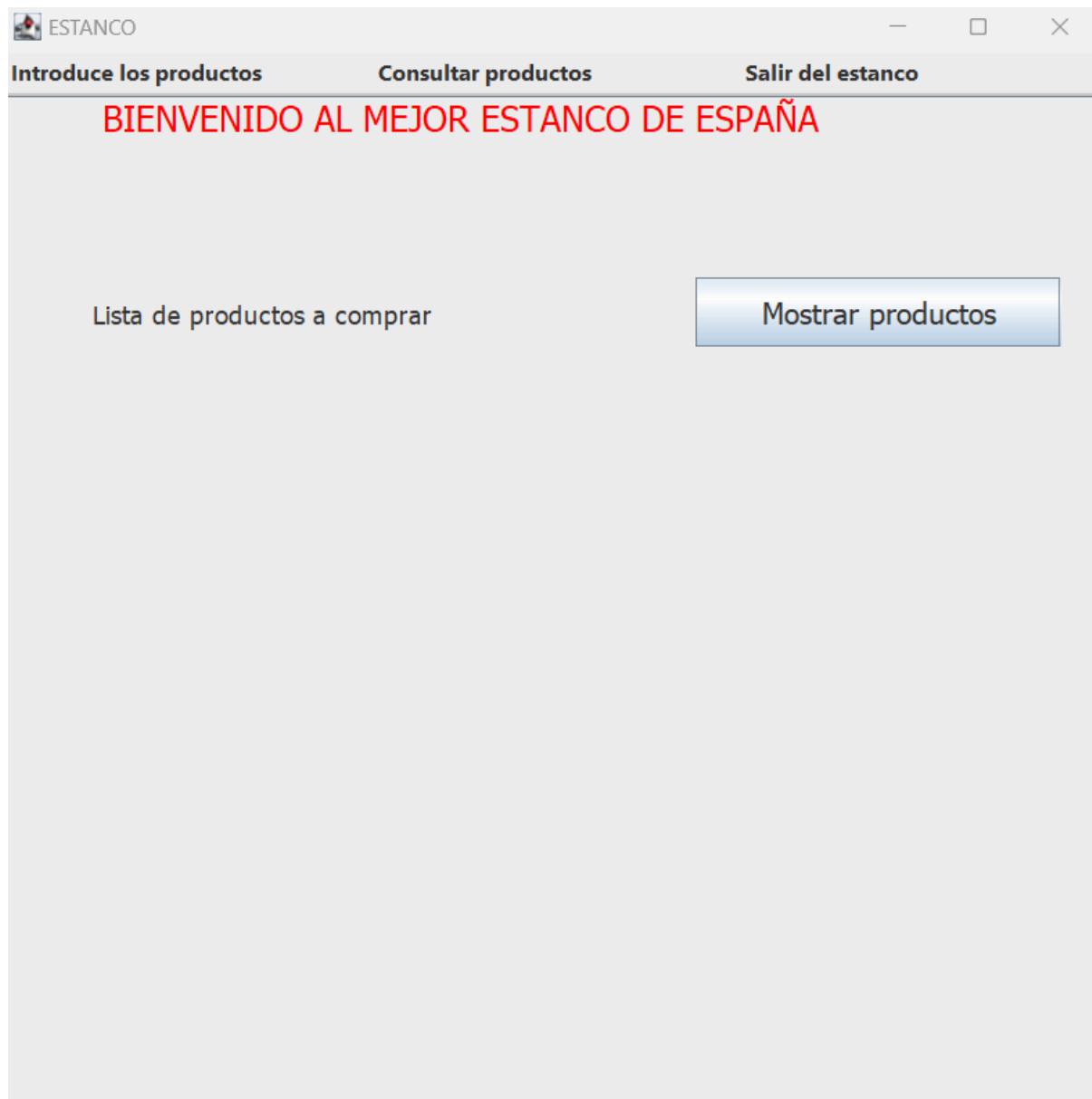
Aceptar

Precio: €

Oferta:

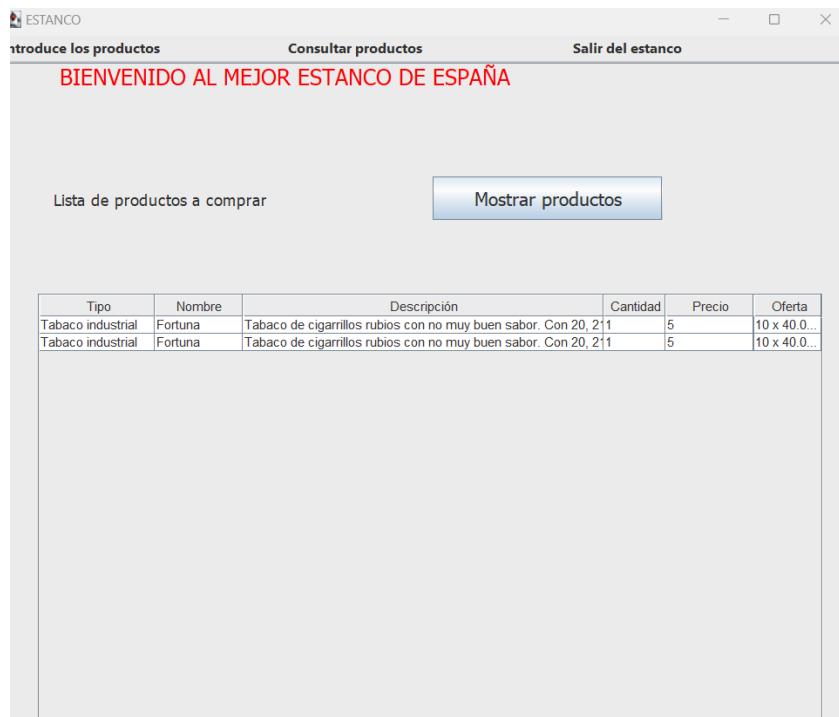
COMPRAR

Una vez se hayan guardado los datos nos iremos a la siguiente interfaz, *Consultar Productos*.



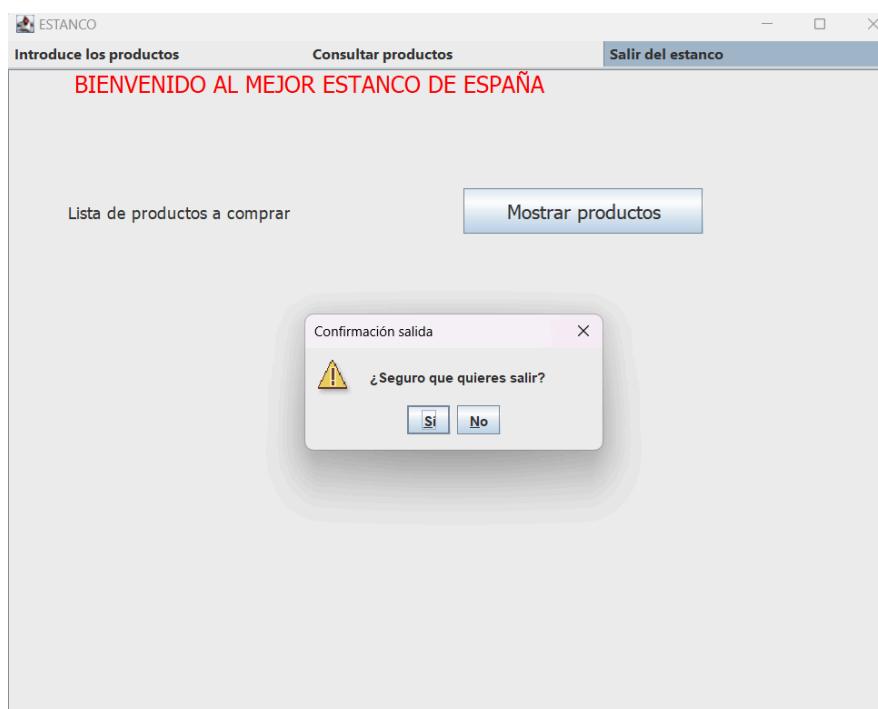
De primeras nos aparece un botón que muestra los productos que hemos comprado.

Y si le damos:



Aparece lo que hemos decidido comprar. El problema es que sale doble y no entiendo por qué.

Por último, en Submenú *Salir del estanco*. Al darle nos aparecerá una ventana de notificación preguntandonos si queremos salir de la página. Si contestas NO te quedaras, si contestas SI la página se cerrará.



Y ya hemos terminado con nuestro Estanco.

M3 Implementar funciones eficaces en la solución para gestionar la concurrencia, la seguridad, las autorizaciones de usuarios y la recuperación de datos.

En cuanto a nuestra base de datos, obviamente es necesario cambiar tanto la contraseña como el usuario, aunque más la contraseña. Si puede ser una con minusculas, mayusculas y numeros, mejor que mejor.

En cuanto a Java, al trabajar con GUI, es importante asegurarse de que las actualizaciones de la interfaz de usuario se realicen en el Event Dispatch Thread (EDT). Es posible usar `SwingUtilities.invokeLater` para asegurarte de que las operaciones de GUI se ejecuten en el hilo correcto. Pongamos un ejemplo:

Ejemplo de autenticación y actualización de GUI en EDT

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        if (autenticarUsuario(username, password)) {
            // Usuario autenticado
            if (usuarioTienePermiso(usuarioActual,
Permiso.EJEMPLO)) {
                // Realizar operación permitida
                realizarOperacion();
            } else {
                // Usuario no autorizado
                mostrarMensajeError("No tiene autorización para
realizar esta operación");
            }
        } else {
            // Falló la autenticación
            mostrarMensajeError("Credenciales incorrectas");
        }
    }
});
```

D2 Evaluar la eficacia del diseño y desarrollo del sistema con respecto a los requisitos del cliente y del sistema.

Bueno considero que aunque el sistema no sea 100% efectivo es un muy buen trabajo, además siendo mi primera experiencia haciendo esto.

Creo que he hecho un buen trabajo. Los nombres de clases y métodos son descriptivos, facilitando la comprensión. He utilizado constantes para valores fijos, reduciendo así errores tipográficos. En la Interfaz de Insertar Productos he descomprimido obtenerObjetos para que sea más estructurado visual.

He mejorado en cuanto a Excepciones a lo largo de este trabajo, pero posiblemente puedo dar más. Sigue habiendo oportunidades para dar mensajes más detallados. Posiblemente el diseño no sea el mejor, pero en cuanto a funcionalidad es un 10, menos por la tabla adobe y los valores desordenados. Debo mejorar y lo haré. Pero diría 100% que al cliente le gustara, sin ninguna duda. El código es sólido y sigue prácticas aceptables. Sin embargo, siempre hay espacio para mejoras, especialmente en la gestión de errores y la optimización de la interfaz de usuario.

Por último daré una última revisión antes de entregárselo al Cliente. Tanto a la bbdd, como al acceso a datos y la eficiencia de las consultas e interfaces.

Todo perfecto.

4º PARTE:

L04 Demostrar las herramientas de administración y gestión de sistemas disponibles en la plataforma elegida.

P5 Demostrar las herramientas disponibles en el sistema para supervisar y optimizar el rendimiento del sistema y examinar los registros de auditoría.

Utiliza el sistema que sea para hacer un registro de auditoría cada vez que se efectúa una operación de actualización en la BD. Puedes hacerlo en una tabla de LOGs por medio de triggers o por escribiendo entradas en un fichero de texto, lo que prefieras.

Explica aquí el sistema elegido y demuestra cómo lo estás llevando a cabo.

P6 Demostrar las herramientas disponibles en el sistema para gestionar la seguridad y las autorizaciones.

Explica todas las medidas que estás utilizando para gestionar la seguridad y las autorizaciones. Por ejemplo: estás utilizando varias cuentas de usuario, ya sea definidas en MySQL o gestionadas desde la aplicación.

M4 Evaluar la eficacia de las herramientas de administración y gestión del sistema disponibles en la plataforma, identificando las posibles deficiencias de las herramientas.
Revisa todas las medidas que has adoptado para garantizar la seguridad, la integridad y el rendimiento.

Realiza un análisis crítico e indica las deficiencias del sistema en términos de seguridad, integridad y rendimiento.

D3 Analizar cualquier mejora futura que pueda ser necesaria para garantizar la eficacia continuada del sistema de base de datos.

Indica todas las mejoras que podrías implementar para mejorar la seguridad, la integridad y el rendimiento.

LO4 Demostrar las herramientas de administración y gestión de sistemas disponibles en la plataforma elegida.

Oracle ofrece varias herramientas de administración y gestión de sistemas que facilitan la supervisión, el ajuste de rendimiento y la gestión de bases de datos.

Algunas de esas herramientas son:

Oracle Enterprise Manager (OEM):

Herramienta de gestión integral que proporciona una interfaz gráfica para administrar bases de datos Oracle. Supervisa el rendimiento, realiza ajustes y tareas administrativas, además de gestionar usuarios.

- Monitoreo en tiempo real del rendimiento de la base de datos.
- Administración de usuarios y privilegios.
- Creación y gestión de copias de seguridad.
- Configuración y ajuste de parámetros.

SQL*Plus:

Interfaz de línea de comandos que permite ejecutar comandos SQL y realizar tareas administrativas directamente desde la consola.

- Ejecución de scripts SQL y PL/SQL.
- Gestión de objetos de base de datos.
- Realización de consultas y actualizaciones.

Oracle SQL Developer: → YO HE UTILIZADO ESTA

Herramienta gráfica para el desarrollo y administración de bases de datos. Proporciona un entorno integrado para escribir consultas SQL, desarrollar scripts y administrar objetos de base de datos.

- Desarrollo y ejecución de consultas SQL.
- Creación y modificación de esquemas de base de datos.
- Herramientas visuales para diseñar y administrar tablas.

RECOMENDADA.

Oracle AWR y ADDM :

AWR (*Automatic Workload Repository*) y ADDM (*Automatic Database Diagnostic Monitor*) son componentes de Oracle Diagnostic Pack que ayudan en la gestión del rendimiento.

- AWR recopila datos de rendimiento para análisis.
- ADDM analiza automáticamente problemas de rendimiento y proporciona recomendaciones.

Oracle Data Pump:

Herramienta para la importación y exportación de datos y metadatos entre bases de datos Oracle.

- Transferencia eficiente de grandes volúmenes de datos.
- Copia de seguridad y restauración de bases de datos.

Oracle Cloud Control:

Basada en la web de Enterprise Manager diseñada para entornos en la nube.

- Supervisión y administración centralizadas de bases de datos en la nube.
- Herramientas de gestión de recursos en la nube.

Todas estas herramientas proporcionan una gran variedad de funciones para administrar y optimizar bases de datos Oracle, adaptándose a las necesidades específicas de los administradores y desarrolladores. La elección de la herramienta dependerá de los requisitos, necesidades y preferencias del usuario.

P6 Demostrar las herramientas disponibles en el sistema para gestionar la seguridad y las autorizaciones.

Para demostrar las herramientas disponibles en el sistema, para gestionar la seguridad y las autorizaciones, de una base de datos Oracle. Oracle Database proporciona diversas herramientas y características para administrar la seguridad y las autorizaciones.

Algunos aspectos clave son:

1. Usuarios y Roles:

- Gestión: Utilizando SQL*Plus o SQL Developer, puedes crear, modificar y eliminar usuarios. Asigna roles a los usuarios para simplificar la asignación de privilegios.

Ejemplo:

```
-- Crear usuario  
CREATE USER ejemplo IDENTIFIED BY contraseña;  
  
-- Asignar privilegios y roles al usuario  
GRANT SELECT, INSERT, UPDATE ON tabla TO ejemplo;  
GRANT ROL_1 TO ejemplo;
```

2. Privilegios y Roles:

- Gestión: Oracle permite asignar privilegios directamente a usuarios o a través de roles. Los roles agrupan privilegios para facilitar la administración.

Ejemplo:

```
-- Crear rol  
CREATE ROLE rol_ejemplo;
```

```
-- Asignar privilegios al rol  
GRANT SELECT, INSERT, UPDATE ON tabla TO rol_ejemplo;  
  
-- Asignar rol a usuario  
GRANT rol_ejemplo TO usuario;
```

3. Oracle Virtual Private Database (VPD):

- Permite implementar políticas de seguridad a nivel de fila para restringir el acceso a datos basado en condiciones.

Ejemplo:

```
-- Crear función de política  
CREATE OR REPLACE FUNCTION vpd_policy (schema_p VARCHAR2,  
table_p VARCHAR2)  
RETURN VARCHAR2 AS...
```

4. Oracle Label Security:

- Permite controlar el acceso a datos basándose en etiquetas de seguridad. Útil en entornos con requisitos de seguridad avanzados.
- Se configura mediante Oracle Enterprise Manager o SQL*Plus con comandos específicos.

5. Oracle Audit Vault and Database Firewall:

- Proporciona funciones avanzadas de auditoría y monitoreo. Puedes definir políticas de auditoría para registrar actividades específicas.
- Se configura a través de la consola de administración de Oracle Audit Vault.

6. Oracle Database Vault:

- Ofrece controles adicionales para reforzar la seguridad. Permite definir reglas de acceso y autorización personalizadas.
- Se configura mediante Oracle Enterprise Manager o scripts SQL específicos.

Estas herramientas y características de Oracle Database proporcionan un conjunto robusto de opciones para gestionar la seguridad y las autorizaciones. La elección de la herramienta específica dependerá de los requisitos de seguridad y de la complejidad del entorno de la base de datos.

D3 Analizar cualquier mejora futura que pueda ser necesaria para garantizar la eficacia continuada del sistema de base de datos.

El análisis para mejorar la eficacia de los sistemas de base de datos es crucial para mantener un entorno eficiente y seguro.

1. Rendimiento y Escalabilidad:

Análisis:

- Evaluar el rendimiento actual del sistema, identificar cuellos de botella y áreas de mejora.
- Considerar el crecimiento futuro de la carga de trabajo y asegurar que el sistema sea escalable.

Mejoras Posibles:

- Ajustes de configuración para optimizar consultas y procesos.
- Implementación de particionamiento de tablas para mejorar el rendimiento en grandes conjuntos de datos.
- Evaluación de la posibilidad de actualización de hardware para mejorar la capacidad.

2. Seguridad y Auditoría:

Análisis:

- Revisar la eficacia de las medidas de seguridad actuales y las políticas de auditoría.
- Identificar posibles amenazas y vulnerabilidades emergentes.

Mejoras Posibles:

- Actualización regular de políticas de seguridad y parches.
- Implementación de soluciones de cifrado avanzadas.
- Integración de herramientas de seguridad adicionales para la detección y prevención de amenazas.

3. Resiliencia y Recuperación ante Desastres:

Análisis:

- Evaluar la capacidad actual para la recuperación ante desastres y la resistencia a fallos.
- Identificar posibles puntos de falla y su impacto en la continuidad del negocio.

Mejoras Posibles:

- Revisión y mejora de estrategias de respaldo y recuperación.
- Implementación de soluciones de alta disponibilidad y replicación de datos.
- Desarrollo de planes de contingencia detallados.

4. Automatización de Tareas Administrativas:

Análisis:

- Identificar tareas administrativas manuales propensas a errores.
- Evaluar la eficiencia de la automatización actual.

Mejoras Posibles:

- Implementación de scripts y procedimientos almacenados para tareas repetitivas.

- Exploración de herramientas de automatización de administración de bases de datos.

5. Integración de Tecnologías Emergentes:

Análisis:

- Mantenerse informado sobre las nuevas tecnologías y tendencias en bases de datos.
- Evaluar la relevancia de tecnologías emergentes para las necesidades comerciales.

Mejoras Posibles:

- Explorar tecnologías como contenedores, orquestación de contenedores y arquitecturas sin servidor.
- Evaluar el impacto de la inteligencia artificial y el aprendizaje automático en la gestión de bases de datos.

6. Capacitación y Desarrollo del Personal:

Análisis:

- Evaluar el nivel de habilidades del personal de administración de bases de datos.
- Identificar áreas de mejora y actualización de conocimientos.

Mejoras Posibles:

- Programas de capacitación continua para mantener al personal actualizado.
- Certificaciones relevantes para la plataforma de base de datos utilizada.