Argentina programa

DATOS Y BASES DE DATOS

Datos

Los datos pueden ser cualquier cosa, desde números, letras, palabras, imágenes, sonidos, vídeos, hasta combinaciones de estos elementos.

Las computadoras son máquinas que procesan datos, esto significa que toman datos de entrada, efectúan operaciones sobre ellos, obteniéndose datos intermedios y, por último, devuelven datos de salida.

Los datos pueden ser crudos, lo que significa que no han sido procesados ni analizados, o pueden ser datos procesados, que han sido transformados de alguna manera para proporcionar información más útil y significativa. Cuando los datos son valores fijos que no pueden ser alterados por el programa se los llama constantes.

A menudo se almacenan en bases de datos o sistemas de archivos, y se utilizan para ayudar a las empresas y organizaciones a tomar decisiones informadas y a obtener información valiosa sobre sus operaciones y clientes.

Información

La información es el resultado del procesamiento y análisis de datos. Es decir, los datos se convierten en información cuando se les da un contexto y significado para que puedan ser entendidos y utilizados para tomar decisiones.

La información puede ser utilizada para describir, explicar o representar algo, ya sea un proceso, un objeto, un evento, una idea, etc. También puede ser utilizada para comunicar un mensaje o para proporcionar conocimientos.

En la era de la información en la que vivimos, la información se ha convertido en uno de los recursos más valiosos, ya que es utilizada por individuos, empresas y gobiernos para tomar decisiones estratégicas y operativas, para planificar y desarrollar proyectos y para comprender mejor el mundo que nos rodea.

Bases de Datos

Una base de datos es un conjunto organizado y estructurado de datos que se almacenan en un sistema de computadora. Una base de datos puede ser

considerada como un conjunto de archivos relacionados que se utilizan para almacenar, organizar y recuperar datos de manera eficiente y precisa.

Las bases de datos son utilizadas por organizaciones y empresas para almacenar información sobre sus clientes, productos, servicios, proveedores, operaciones y cualquier otra cosa que necesiten gestionar. Las bases de datos también son utilizadas por aplicaciones informáticas y sitios web para almacenar y recuperar información de manera rápida y eficiente.

Las bases de datos pueden ser gestionadas mediante software especializado llamado sistema de gestión de bases de datos o motor de bases de datos (DBMS, por sus siglas en inglés). Los DBMS permiten a los usuarios agregar, eliminar y modificar datos, así como realizar consultas y generar informes basados en la información almacenada en la base de datos.

Como analista QA manual, requerimos en oportunidades de crear conjunto de datos que utilizaremos para ejecutar los casos de prueba o verificar que la información que visualizamos en el aplicativo o sitio web es la correcta. En esta unidad aprenderemos los conceptos básicos de una base de datos relacional y como obtener información directamente desde una base de datos. Veamos primero algunos conceptos más de bases de datos.

Bases de Datos Relacional vs No Relacional

Existen dos tipos principales de bases de datos: las bases de datos relacionales y las bases de datos no relacionales (también conocidas como bases de datos NoSQL). A continuación, se describen brevemente las características de cada uno de estos tipos de bases de datos.

Bases de datos relacionales:

Las bases de datos relacionales se organizan generalmente en tablas que contienen filas y columnas. Las filas representan registros individuales y las columnas representan campos de datos. Cada registro en una tabla tiene un identificador único, llamado clave primaria, que permite una fácil identificación y recuperación de la información. Se basan en el lenguaje de consulta estructurado (SQL) para manipular y consultar los datos almacenados.

Son adecuadas para almacenar datos estructurados y altamente relacionales, como los datos de una empresa, transacciones financieras, registros de clientes, entre otros.

Bases de datos no relacionales:

Utilizan un modelo de datos no estructurado o semi-estructurado, como documentos, gráficos, pares clave-valor, entre otros. No utilizan SQL como lenguaje de consulta, sino que utilizan otros lenguajes o interfaces para manipular los datos.

Tienen una estructura flexible y escalable, que permite agregar y modificar campos y estructuras de datos sin la necesidad de redefinir la estructura completa de la base de datos.

Son adecuadas para almacenar grandes cantidades de datos no estructurados, como los datos de redes sociales, datos de sensores, registros de aplicaciones web, entre otros.

Cada tipo de base de datos tiene sus propias ventajas y desventajas, y la elección de una u otra depende del tipo de datos que se desean almacenar y del propósito para el que se utilizarán.

En este curso veremos en más detalle las bases de datos relacionales.

Motor de Bases de Datos

Un motor de bases de datos es un software que gestiona la organización, almacenamiento, acceso y manipulación de datos en una base de datos. En otras palabras, es el componente central que permite a una base de datos funcionar.

El motor de bases de datos es responsable de recibir y procesar solicitudes de datos de los usuarios y aplicaciones, y de realizar operaciones como la creación, actualización y eliminación de datos en la base de datos. También es responsable de garantizar la integridad de los datos y la seguridad de la base de datos.

Los motores de bases de datos pueden clasificarse en diferentes tipos según su arquitectura, como, por ejemplo:

Bases de datos relacionales: Los motores de bases de datos relacionales son los más comunes y utilizan una estructura de tablas relacionadas entre sí para almacenar y gestionar los datos. Algunos de los motores de bases de datos relacionales más conocidos son:

MySQL: es uno de los motores de bases de datos más populares y ampliamente utilizados en todo el mundo. Es un software de código abierto y se utiliza para aplicaciones web y empresariales.

Oracle Database: es un motor de bases de datos empresarial que se utiliza para aplicaciones empresariales y de grandes volúmenes de datos. Es conocido por su escalabilidad, rendimiento y seguridad.

Microsoft SQL Server: es un motor de bases de datos desarrollado por Microsoft que se utiliza para aplicaciones empresariales y web. Es conocido por su facilidad de uso y herramientas de gestión.

PostgreSQL: es un motor de bases de datos de código abierto que se utiliza para aplicaciones empresariales y web. Es conocido por su robustez, escalabilidad y capacidad para manejar grandes volúmenes de datos.

SQLite: es un motor de bases de datos ligero y autónomo que se utiliza para aplicaciones móviles y web. Es conocido por su facilidad de uso y bajo consumo de recursos.

IBM DB2: es un motor de bases de datos empresarial que se utiliza para aplicaciones empresariales y de grandes volúmenes de datos. Es conocido por su escalabilidad, rendimiento y seguridad.

SAP Sybase ASE: es un motor de bases de datos empresarial que se utiliza para aplicaciones empresariales y de grandes volúmenes de datos. Es conocido por su rendimiento, escalabilidad y capacidad para manejar grandes volúmenes de datos.

MariaDB: es un motor de bases de datos de código abierto que se utiliza para aplicaciones web y empresariales. Es una bifurcación de MySQL y se ha vuelto popular debido a su compatibilidad con MySQL y su énfasis en la seguridad.

Cada uno de estos motores de bases de datos tiene sus propias características y beneficios, y la elección del motor de base de datos adecuado depende de los requisitos específicos de cada proyecto.

Bases de datos NoSQL: Los motores de bases de datos NoSQL utilizan estructuras de datos no relacionales para almacenar y gestionar los datos. Algunos motores de bases de datos NoSQL más conocidos son:

MongoDB: Es un motor de bases de datos NoSQL de tipo documental, que utiliza documentos en formato JSON para almacenar y recuperar datos.

Cassandra: Es un motor de bases de datos NoSQL de tipo clave-valor, que se utiliza para almacenar grandes volúmenes de datos distribuidos en múltiples servidores.

Neo4j: Es un motor de bases de datos NoSQL de tipo grafo, que se utiliza para almacenar y recuperar datos con relaciones complejas.

Redis: Es un motor de bases de datos NoSQL de tipo clave-valor, que se utiliza para almacenar datos en memoria y recuperarlos rápidamente.

Amazon DynamoDB: Es un motor de bases de datos NoSQL de tipo clave-valor, que se utiliza en la nube de Amazon Web Services para almacenar datos escalables y de alta disponibilidad.

Bases de datos en memoria: Los motores de bases de datos en memoria almacenan los datos directamente en la memoria del sistema, en lugar de en un disco duro, lo que permite una mayor velocidad de acceso a los datos.

Bases de datos distribuidas: Los motores de bases de datos distribuidas permiten que la base de datos se distribuya en múltiples servidores para mejorar el rendimiento y la disponibilidad.

En resumen, un motor de bases de datos es un software que gestiona la organización, almacenamiento, acceso y manipulación de datos en una base de datos. Los motores de bases de datos pueden clasificarse en diferentes tipos según su arquitectura, y cada tipo tiene sus propias características y ventajas.

Conceptos característicos de Bases de Datos Relacionales

Tabla

Es la estructura básica en la base de datos relacional, y representa una colección de datos organizados en filas y columnas. Cada tabla tiene un nombre único y está formada por campos o columnas, que representan las diferentes categorías de datos, y por registros o filas, que contienen los valores de los campos. Estamos acostumbrados a representar datos de esta forma si utilizamos por ejemplo una planilla Excel:

	A	B	C	D	E	F	G
1							
2							
3							
4		N° de Factura	Fecha	Cliente	Producto	Total Venta	
5		1	8/1/2023	Mirtha	Limones	$ 120,00	
6		2	8/1/2023	Laura	Pomelos	$ 200,00	
7		3	15/1/2023	Carolina	Bananas	$ 400,00	
8		4	29/1/2023	Los Chilakiles S.A	Lechuga	$ 150,00	
9		5	12/3/2023	Sansho SRL	Tomates	$ 300,00	
10		6	2/4/2023	LaliL	Naranjas	$ 250,00	
11							
12							
13							
14							
15							

Este ejemplo podría representar la tabla ventas donde posee 5 campos, representado en las 5 columnas, cada una un dato diferente: Numero de factura, fecha, cliente, producto, total de la venta. Las filas son los registros donde se encuentra cada una de las ventas y la información claramente está relacionada entre sí. Ejemplo: Posee 6 registros, si leemos la información del primer registro podemos decir: La factura 1 se efectuó el 8-1-2023 por Mirtha, compro limones por un total de 120 pesos.

Es importante mencionar que una base de datos contendrá varias tablas, tantas como información necesite almacenar. Las tabas se relacionan entre sí, en breve veremos en mayor detalle este concepto.

Clave primaria

Es un campo o conjunto de campos que identifican de forma única cada registro en una tabla. Cada tabla tiene una clave primaria, que permite la identificación y la vinculación con otras tablas. En el ejemplo podemos decir que la clave primara es el número de factura, no se repite e identifica de forma única a un solo registro de la tabla ventas.

La clave primaria de las tablas las define quien diseña las bases de datos y es fundamental conocerlo si nos encargamos de obtener la información desde las tablas para crear los conjuntos de datos.

Clave foránea

Es un campo o conjunto de campos que se utiliza para vincular dos o más tablas en una base de datos relacional. La clave foránea de una tabla se corresponde con la clave primaria de otra tabla, y se utiliza para establecer relaciones entre los datos almacenados en ambas tablas.

Índice

Es una estructura de datos que se utiliza para acelerar el acceso a los datos almacenados en una tabla. Los índices se crean en uno o más campos de una tabla, y permiten la búsqueda rápida de datos utilizando valores específicos.

Relaciones entre tablas

Las relaciones entre tablas en una base de datos relacional se establecen mediante el uso de claves primarias y claves foráneas. Recordemos que una clave primaria es un campo o conjunto de campos que identifican de forma única cada registro en una tabla. Por otro lado, una clave foránea es un campo que hace referencia a una clave primaria en otra tabla.

Existen tres tipos principales de relaciones entre tablas en una base de datos relacional:

Relación uno a uno (1:1): En una relación uno a uno, cada registro de una tabla está relacionado con un único registro de otra tabla y viceversa. Por ejemplo, si tenemos una tabla de "Clientes" y una tabla de "Direcciones", cada cliente puede tener una única dirección y cada dirección puede pertenecer a un único cliente. En este caso, la clave primaria de la tabla "Clientes" se convierte en la clave foránea en la tabla "Direcciones".

Relación uno a muchos (1:N): En una relación uno a muchos, cada registro de una tabla puede estar relacionado con varios registros de otra tabla, pero cada registro de la segunda tabla solo puede estar relacionado con un registro de la primera tabla. Por ejemplo, si tenemos una tabla de "Departamentos" y una tabla de "Empleados", cada departamento puede tener varios empleados, pero cada empleado solo puede trabajar en un único departamento. En este caso, la clave primaria de la tabla "Departamentos" se convierte en la clave foránea en la tabla "Empleados".

Relación muchos a muchos (N:N): En una relación muchos a muchos, cada registro de una tabla puede estar relacionado con varios registros de otra tabla y viceversa. Por ejemplo, si tenemos una tabla de "Estudiantes" y una tabla de "Cursos", cada estudiante puede estar matriculado en varios cursos y cada curso puede tener varios estudiantes matriculados. En este caso, se necesita una tabla intermedia que contenga las claves primarias de ambas tablas (tabla "Matrículas"), y ambas claves primarias se convierten en claves foráneas en la tabla intermedia.

En resumen, las relaciones entre tablas en una base de datos relacional se establecen mediante el uso de claves primarias y foráneas, y pueden ser uno a uno, uno a muchos o muchos a muchos. Es importante diseñar las relaciones adecuadamente para evitar redundancia y asegurar la integridad de los datos.

Restricciones de integridad

Las restricciones de integridad en una base de datos relacional son reglas o condiciones que se imponen en los datos almacenados en la base de datos para garantizar que sean precisos y consistentes. Estas restricciones se utilizan para mantener la integridad de los datos en la base de datos y asegurar que se mantengan en un estado válido y coherente.

Existen diferentes tipos de restricciones de integridad que se pueden aplicar en una base de datos relacional, entre ellas:

Restricciones de clave primaria: Estas restricciones aseguran que cada fila de una tabla tenga un valor único en una o varias columnas específicas. Como vimos, la clave primaria es una columna o conjunto de columnas que identifica de manera única cada fila en una tabla y es necesaria para garantizar la integridad referencial.

Restricciones de clave externa: Estas restricciones garantizan que los valores en una columna específica de una tabla coincidan con los valores de la columna de clave primaria en otra tabla. Esto ayuda a mantener la integridad referencial entre las tablas y evita inconsistencias en los datos.

Restricciones de unicidad: Estas restricciones aseguran que los valores en una columna específica de una tabla sean únicos, pero no necesariamente que la columna sea una clave primaria.

Restricciones de comprobación: Estas restricciones verifican que los valores en una columna específica cumplan con una determinada condición o regla. Esto puede ser útil para garantizar que los valores ingresados en una tabla cumplan con ciertas especificaciones o reglas de negocio.

En resumen, las restricciones de integridad son reglas o condiciones que se imponen en los datos almacenados en una base de datos relacional para garantizar que sean precisos y consistentes. Estas restricciones incluyen restricciones de clave primaria, restricciones de clave externa, restricciones de unicidad y restricciones de comprobación, entre otras.

Consulta

Es una solicitud para acceder a los datos almacenados en una o más tablas de una base de datos relacional. Las consultas se realizan utilizando el lenguaje SQL, y permiten recuperar, modificar y eliminar datos de la base de datos. En este curso veremos los conceptos básicos y cómo podemos modificar las consultas para obtener diferentes conjuntos de datos.

Vistas

En una base de datos relacional, una vista es una tabla virtual que se crea a partir de una o varias tablas existentes. A diferencia de una tabla física, una vista no almacena datos por sí misma, sino que muestra una vista personalizada de los datos almacenados en las tablas subyacentes.

La vista puede ser creada con una consulta SQL que selecciona y combina los datos de una o varias tablas, y luego puede ser utilizada como si fuera una tabla física en las consultas posteriores. Las vistas pueden tener restricciones adicionales sobre los datos que muestran, lo que permite restringir el acceso a datos sensibles o complejos.

Entre las principales ventajas de las vistas se incluyen:

Simplificación de las consultas: Las vistas pueden simplificar las consultas a la base de datos al proporcionar una vista personalizada y estructurada de los datos, lo que puede facilitar la extracción de la información requerida.

Seguridad: Las vistas pueden ayudar a proteger la información sensible al restringir el acceso a datos confidenciales o complejos.

Reducción de la redundancia: Las vistas pueden reducir la redundancia de datos al evitar la duplicación de información en varias tablas.

Mejora del rendimiento: Las vistas pueden mejorar el rendimiento de las consultas al permitir la recuperación de datos previamente calculados y almacenados en caché.

En resumen, una vista es una tabla virtual que se crea a partir de una o varias tablas existentes y que proporciona una vista personalizada de los datos almacenados en la base de datos. Las vistas son una herramienta útil para simplificar las consultas, mejorar la seguridad, reducir la redundancia de datos y mejorar el rendimiento.

SQL

SQL (Structured Query Language) es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales. Fue desarrollado en los años 70 y es utilizado en todo el mundo por desarrolladores de software, administradores de bases de datos y analistas de datos.

SQL permite realizar una variedad de tareas, como crear bases de datos y tablas, insertar, actualizar y eliminar datos, y realizar consultas complejas para extraer información de la base de datos. Además, SQL permite definir relaciones entre tablas, establecer restricciones de integridad, crear índices y vistas, y gestionar permisos de acceso a la información.

El lenguaje SQL se divide en varias categorías, incluyendo:

DDL (Data Definition Language): Permite definir la estructura de la base de datos, como crear tablas, definir claves primarias y foráneas, crear índices, etc.

DML (Data Manipulation Language): Permite manipular los datos almacenados en la base de datos, como insertar, actualizar y eliminar registros.

DQL (Data Query Language): Permite realizar consultas a la base de datos para extraer información.

DCL (Data Control Language): Permite definir permisos de acceso a la base de datos y controlar la seguridad.

En resumen, SQL es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales, permitiendo la definición de estructuras, manipulación de datos y consultas complejas. Es ampliamente utilizado en el mundo de la tecnología y es esencial para el manejo de grandes cantidades de datos en una organización.

Diseñar una base de datos relacional

Si bien no nos encargamos de crear las bases de datos, conocer como es el proceso para crearlas nos permitirá comprender mejor como consultamos los datos en ellas.

Diseñar una base de datos relacional implica seguir una serie de pasos que permiten identificar las tablas, campos y relaciones entre tablas que conforman el modelo de datos. Básicamente podemos resumir los pasos para crear una base de datos de la siguiente forma:

Identificar las tablas: Primero identificamos las tablas donde almacenaremos los datos relacionados y que tienen una existencia independiente y son relevantes para el negocio. Por ejemplo, en una base de datos para una tienda en línea, las tablas podrían ser "Clientes", "Productos", "Órdenes de compra", etc.

Definir los campos de las tablas: luego definimos los campos que son las características o propiedades de las tablas, como vimos anteriormente, representados en columnas. Por ejemplo, para la tabla "Productos", los campos podrían ser "nombre", "precio", "descripción", etc.

Identificar las relaciones entre las tablas: es importante identificar las relaciones, o sea, las conexiones que existen entre las tablas. Por ejemplo, en una tienda en línea, la relación entre "Clientes" y "Órdenes de compra" es que un cliente puede hacer varias órdenes de compra y una orden de compra está asociada a un cliente.

Identificar las claves primarias: Analizamos a detalle los datos que contiene una tabla para identificar correctamente una clave primaria, recordemos que es un campo o conjunto de campos que identifican de forma única una entidad. Por ejemplo, en la entidad "Clientes", la clave primaria podría ser el número de identificación del cliente.

Definir las restricciones de integridad: definimos las restricciones de integridad son reglas que se aplican a los datos para asegurar su consistencia y validez. Por ejemplo, una restricción de integridad podría ser que el precio de un producto no puede ser negativo.

Normalizar las tablas: La normalización es un proceso que permite eliminar la redundancia de los datos y optimizar la estructura de las tablas. Para ello, se deben identificar las dependencias funcionales entre los atributos y dividir las tablas en aquellas que cumplan con las reglas de normalización. Este

proceso lo realizan los diseñadores de bases de datos, no lo veremos en este curso.

Crear el diagrama entidad-relación (ER): El diagrama ER es una representación gráfica del modelo de datos que muestra las tablas, relaciones y campos, así como las claves primarias y restricciones de integridad. Aprenderemos a leer los diagramas entidad relación.

Crear las tablas y definir las relaciones: Una vez que se tiene el diagrama ER, se pueden crear las tablas en la base de datos y definir las relaciones entre ellas.

Añadir datos: Finalmente, se pueden añadir los datos a la base de datos mediante la inserción de registros en las tablas.

En resumen, para diseñar una base de datos relacional es necesario identificar las tablas, relaciones y campos, definir las claves primarias y restricciones de integridad, normalizar las tablas, crear el diagrama ER, crear las tablas y definir las relaciones, y añadir los datos.

Como analistas QA manual, nos interesa ser capaces de interpretar un diagrama entidad relación, entender una consulta SQL que nos permita acceder a la información y manipular las condiciones de la consulta de forma que logremos obtener los diferentes conjuntos de datos.

Realizando consultas a una base de datos relacional

Para realizar una consulta a una base de datos relacional, se puede utilizar el lenguaje SQL (Structured Query Language). Como vimos anteriormente, nos permite realizar una amplia variedad de consultas y operaciones.

Para realizar una consulta a una base de datos relacional utilizando SQL, se deben seguir los siguientes pasos generales:

Seleccionar la tabla o tablas de la base de datos de las que se desean recuperar datos.

Escribir una consulta SQL que especifique los datos que se desean recuperar. Esto se logra utilizando la cláusula SELECT de SQL, que permite seleccionar columnas específicas de una tabla o tablas, o bien utilizar la cláusula * para seleccionar todas las columnas de una tabla.

Especificar cualquier criterio de selección utilizando la cláusula WHERE de SQL. Esta cláusula permite filtrar los datos que se recuperan de la base de datos según ciertos criterios, como valores específicos, rangos de valores o combinaciones de condiciones.

Si se desea, se pueden agregar cláusulas adicionales como GROUP BY, ORDER BY y JOIN para especificar el orden en que se desean los datos, agruparlos y combinarlos de diferentes maneras.

Ejecutar la consulta SQL en el cliente de base de datos. Esto recuperará los datos seleccionados de la base de datos y los mostrará en la pantalla.

Aquí hay un ejemplo simple de una consulta SQL para recuperar todos los registros de una tabla "usuarios" que tengan un campo "nombre" igual a "Juan":

```sql
SELECT * FROM usuarios WHERE nombre = 'Juan';
```

Este es solo un ejemplo básico, pero SQL es un lenguaje muy poderoso y versátil que puede ser utilizado para realizar una amplia variedad de consultas y operaciones en una base de datos relacional.

Ejemplos de consultas a bases de datos SQL

Veamos las cláusulas básicas y ejemplo de cada una de ellas

Cláusula SELECT

La cláusula SELECT se utiliza para seleccionar columnas específicas de una tabla, podemos mencionar los nombres de los campos o utilizar el asterisco para representar todos los campos de una tabla, veremos ejemplos de SELECT junto con la cláusula FROM

Cláusula FROM

La cláusula FROM se utiliza para especificar la tabla o tablas de las que se van a seleccionar los datos.

Ejemplo 1: Seleccionar todos los registros de la tabla "usuarios".

```sql
SELECT * FROM usuarios;
```

En caso de necesitar información de más de una tabla, se separan por comas y además necesitamos utilizar la cláusula WHERE, veremos mas ejemplos a continuación.

Ejemplo 2: Seleccionar solo la columna "nombre" de la tabla "clientes".

```sql
SELECT nombre FROM clientes;
```

Ejemplo 3: Seleccionar las columnas "nombre" y "edad" de la tabla "empleados"

```sql
SELECT nombre, edad FROM empleados;
```

Cláusula WHERE

Utilizando la cláusula WHERE, podemos filtrar los registros de una tabla y devolver solo aquellos que cumplen con las condiciones especificadas en la consulta.

Ejemplo 1: Seleccionar todos los registros de la tabla "usuarios" donde el valor de la columna "edad" es igual a 30.

```sql
SELECT * FROM usuarios WHERE edad = 30;
```

Ejemplo 2: Seleccionar los registros de la tabla "pedidos" donde el valor de la columna "fecha" es posterior al 1 de enero de 2022.

```sql
SELECT * FROM pedidos WHERE fecha > '2022-01-01';
```

Ejemplo 3: Seleccionar los registros de la tabla "productos" donde el valor de la columna "precio" es mayor a 100 y menor a 200.

```sql
SELECT * FROM productos WHERE precio > 100 AND precio < 200;
```

Ejemplo 4: Seleccionar los registros de la tabla "clientes" donde el valor de la columna "nombre" contiene la cadena "González".

```sql
SELECT * FROM clientes WHERE nombre LIKE '%González%';
```

Cláusula ORDER BY

Utilizando la cláusula ORDER BY, en ocasiones necesitaremos darle un orden determinado a los datos que obtenemos de la base de datos, por lo que requerimos indicarle en que orden deseamos visualizar la información, veamos algunos ejemplos:

Ejemplo 1: Seleccionar todos los registros de la tabla "usuarios" ordenados por la columna "nombre" de forma ascendente.

```sql
SELECT * FROM usuarios ORDER BY nombre ASC;
```

Ejemplo 2: Seleccionar los registros de la tabla "pedidos" ordenados por la columna "fecha" de forma descendente.

```sql
SELECT * FROM pedidos ORDER BY fecha DESC;
```

Ejemplo 3: Seleccionar los registros de la tabla "productos" ordenados por la columna "precio" de forma ascendente y luego por la columna "nombre" de forma descendente.

```sql
SELECT * FROM productos ORDER BY precio ASC, nombre DESC;
```

Ejemplo 4: Seleccionar los registros de la tabla "clientes" ordenados alfabéticamente por la columna "ciudad" y luego por la columna "nombre" de forma ascendente.

```sql
SELECT * FROM clientes ORDER BY ciudad, nombre ASC;
```