ChatGPT

Las Bases de Datos Relacionales: Rol, Características y Elementos Fundamentales

El rol de las bases de datos relacionales en la organización

Ilustración conceptual de la gestión de datos: una base de datos central (círculo) conectada con terminales que representan el análisis de información en una organización. Las bases de datos relacionales cumplen un papel fundamental en las empresas modernas al **centralizar y gestionar la información de forma eficiente**. No se trata solo de un repositorio pasivo, sino de un sistema activo que **almacena, procesa y recupera datos de manera estructurada**, garantizando su integridad, seguridad y disponibilidad [1]. Esto permite que en una organización se pueda **acceder rápidamente a información clave** sobre clientes, inventarios, ventas, operaciones, etc., lo que **facilita la toma de decisiones informadas y oportunas** [2]. En la era digital, la información es uno de los activos más valiosos; contar con una base de datos sólida y bien estructurada se ha vuelto indispensable para convertir datos en conocimiento útil para el negocio.

Entre los principales **beneficios** que ofrece una base de datos relacional a nivel empresarial se pueden mencionar [3]:

- **Unificación de datos**: Agrupa y almacena todos los datos de la empresa en un único lugar centralizado, evitando dispersiones [4].
- **Colaboración**: Facilita que los distintos usuarios o departamentos compartan y accedan a los mismos datos de forma controlada [4].
- **Reducción de redundancia**: Elimina duplicaciones innecesarias y mejora la organización de la información, manteniendo un solo registro para cada dato [5].
- **Visión 360° del cliente**: Permite visualizar rápidamente todos los datos relevantes de un cliente (interacciones, ventas, contactos, etc.) en un mismo sistema [6].

- **Integración de procesos**: Conecta datos de distintas operaciones (ventas, facturación, logística, etc.) para obtener una visión integrada del negocio [6].

Cuando una base de datos se **gestiona adecuadamente**, la organización obtiene ventajas notables: aumentan la eficacia y agilidad en las operaciones y se mejora la seguridad de los datos, lo que redunda en mayor productividad [7]. Las decisiones estratégicas pasan a basarse en datos reales en lugar de suposiciones; de hecho, las compañías más competitivas **basan sus decisiones en datos verificables** y análisis rigurosos, identificando patrones de comportamiento, evaluando rentabilidad y detectando áreas de mejora [8]. Sin datos confiables, cualquier estrategia sería un experimento incierto, por lo que la **gestión de bases de datos es hoy un componente esencial** de la planificación y crecimiento sostenible de las organizaciones [9].

Otro aspecto crucial del rol de las bases de datos es la **protección y seguridad de la información** corporativa. En un entorno donde los datos sensibles (de clientes, transacciones, etc.) deben resguardarse cuidadosamente, una base de datos bien diseñada incluye mecanismos de seguridad (autenticación de usuarios, permisos, cifrado) y copias de respaldo. Esto ayuda a prevenir pérdidas de información, filtraciones o accesos no autorizados [10]. En suma, una base de datos relacional en la empresa actúa como **el corazón del sistema de información**, proporcionando *accesibilidad mejorada* a los datos, *seguridad reforzada*, *integridad* y *consistencia* en la información compartida, reduciendo redundancias y permitiendo escalar el manejo de datos conforme crece el negocio [11] [12]. Todo ello convierte a las bases de datos en un pilar para la eficiencia, la toma de decisiones y la ventaja competitiva en la era de la transformación digital.

Características de un SGBD relacional para la gestión de la información

Una **base de datos relacional** se define como un conjunto de datos organizados en tablas interrelacionadas, administrados por un sistema gestor (SGBD) que sigue el modelo relacional. A continuación se presentan las **características clave** de un Sistema de Gestión de Bases de Datos Relacionales (SGBDR) orientado a la eficaz gestión de la información:

- **Organización en tablas**: Los datos se estructuran en *tablas* formadas por filas (registros) y columnas (campos o atributos). Cada tabla representa una entidad lógica (por ejemplo, clientes, productos, ventas) y cada fila almacena una instancia o registro de dicha entidad [13]. Esta disposición tabular facilita la comprensión y manipulación de grandes volúmenes de información.

- **Relaciones mediante claves**: Las tablas pueden vincularse entre sí mediante *claves primarias* (identificadores únicos de cada registro) y *claves foráneas* (referencias a claves primarias en otras tablas). Estas relaciones permiten establecer vínculos consistentes entre datos de tablas diferentes (por ejemplo, relacionar pedidos con clientes) y garantizan la **integridad referencial**, evitando datos huérfanos o inconsistentes [14] [15]. Las restricciones relacionales aseguran que no haya incongruencias, de modo que si un registro depende de otro, no pueda existir si su referente falta o ha sido eliminado.

- **Lenguaje de consulta estándar (SQL)**: El **Structured Query Language (SQL)** es el lenguaje declarado estándar para interactuar con bases de datos relacionales. Permite definir la estructura de la base de datos (DDL), manipular los datos (DML) y consultar información de forma declarativa mediante sentencias (`SELECT`, `INSERT`, `UPDATE`, `DELETE`, etc.) [16].

Gracias a SQL, usuarios y aplicaciones pueden realizar desde operaciones simples hasta consultas muy complejas para extraer información valiosa de los datos.

- **Transacciones ACID**: Los SGBD relacionales gestionan operaciones en forma de *transacciones* que cumplen las propiedades **ACID** – *Atomicidad, Consistencia, Aislamiento y Durabilidad*. Esto significa que las series de cambios en la base de datos se ejecutan con garantía de "todo o nada" (atomicidad), manteniendo el estado consistente de los datos ante cada operación (consistencia), aislando los efectos de transacciones concurrentes para evitar interferencias (aislamiento) y haciendo permanentes los cambios una vez confirmados, incluso frente a fallos del sistema (durabilidad) [17] [18]. Estas propiedades aseguran la **integridad de las transacciones** y evitan corrupción o pérdida de datos aun en escenarios de error.

- **Acceso multiusuario y control de concurrencia**: Un SGBD permite que múltiples usuarios y aplicaciones accedan **simultáneamente** a la base de datos sin conflicto. Para lograrlo, implementa mecanismos de *control de concurrencia* (como bloqueos o *locking* de registros, niveles de aislamiento de transacciones, etc.) que **evitan inconsistencias cuando varios usuarios leen o escriben datos al mismo tiempo** [19]. De esta forma, varios departamentos o procesos pueden trabajar sobre la misma información compartida en paralelo, manteniendo la coherencia de los datos.

- **Seguridad de los datos**: Los sistemas de bases de datos incluyen herramientas para proteger la información sensible. Esto abarca **control de accesos y privilegios de usuario** (solo personal autorizado puede ver o modificar determinados datos), autenticación robusta, y a veces cifrado de datos almacenados o en tránsito. El SGBD garantiza que los datos confidenciales estén resguardados contra accesos no autorizados y cumple con políticas o normativas de privacidad cuando es necesario [20]. La gestión centralizada de usuarios y permisos permite un acceso controlado y registra actividades (auditoría), fortaleciendo la seguridad de la información corporativa.

- **Escalabilidad y rendimiento**: Las bases de datos relacionales están diseñadas para **manejar grandes volúmenes de datos** y altas cargas de trabajo a medida que la empresa crece. Un buen SGBD puede escalar verticalmente (mejor hardware) u horizontalmente (distribución en múltiples servidores) según la necesidad [21]. Además, incorpora optimizaciones como **índices**, *cachés* de consultas y planes de ejecución optimizados para acelerar el acceso a datos. La estandarización en SQL facilita también la migración o crecimiento, ya que el conocimiento es transferible entre distintos sistemas [22]. En entornos empresariales, SGBD avanzados logran atender miles de transacciones por segundo manteniendo tiempos de respuesta adecuados.

- **Copias de seguridad y recuperación**: Una característica indispensable es la capacidad de realizar **backups periódicos** de la base de datos y de recuperar los datos en caso de fallos o pérdidas. Los SGBD suelen proveer utilidades para volcar datos a copias de seguridad (respaldos completos, incrementales) y para restaurarlos de forma confiable. Esto asegura la continuidad del negocio y evita desastres ante fallos de hardware o errores humanos [23]. Junto con esto, muchos sistemas incluyen funciones de **recuperación ante fallos** (journaling, logging de transacciones) que permiten restablecer la base de datos al último estado consistente tras un corte de energía u otro incidente.

En resumen, un SGBD relacional ofrece una plataforma robusta, segura y consistente para la gestión de la información. **Garantiza la integridad de los datos**, admite la interacción concurrente de muchos usuarios, protege la información sensible y proporciona un lenguaje unificado (SQL) para explotar los

datos. Estas características explican por qué, pese al surgimiento de otros modelos, las bases de datos relacionales sigan siendo la piedra angular de la mayoría de los sistemas de información corporativos.

Alternativas de bases de datos más utilizadas en la industria

En el mercado existen diversas **implementaciones de sistemas de bases de datos relacionales** (RDBMS) desarrolladas por distintos proveedores, cada una con sus particularidades, pero todas basadas en principios similares. A continuación se destacan algunas de las **alternativas más utilizadas en la industria** y sus características generales:

- **MySQL / MariaDB** – Es una de las bases de datos relacionales **más populares** de código abierto. MySQL ha sido ampliamente adoptada para aplicaciones web y empresariales por su **flexibilidad, escalabilidad y facilidad de uso** [24]. Grandes plataformas en Internet la han utilizado para manejar volúmenes masivos de información. MariaDB es un fork (derivación) compatible con MySQL, también open source, con mejoras de rendimiento y que mantiene la filosofía libre tras la adquisición de MySQL por Oracle. Ambas son muy apreciadas por su comunidad activa y por ser multiplataforma (Windows, Linux, macOS).

- **PostgreSQL** – Otro sistema relacional de código abierto, reconocido por ser **extremadamente robusto y orientado a estándares**. PostgreSQL soporta consultas SQL avanzadas y una variedad de tipos de datos, incluyendo JSON para datos semiestructurados [25]. Es conocido por su énfasis en la integridad y coherencia de los datos (cumple estrictamente ACID) y por su capacidad de extenderse (permite definir tipos de datos y funciones personalizadas). Debido a su fiabilidad, PostgreSQL se utiliza frecuentemente en entornos empresariales críticos que requieren alto rendimiento y operaciones complejas.

- **Oracle Database** – Es uno de los RDBMS **comerciales más avanzados y extendidos** en grandes corporaciones. Oracle es reconocido por su **alto rendimiento, escalabilidad y seguridad**, siendo capaz de gestionar enormes cantidades de datos y transacciones con mínimas caídas [26]. Ofrece características avanzadas para alta disponibilidad (clústeres, replicación), robustez en transacciones y amplias herramientas de administración. Muchas organizaciones financieras, gubernamentales y corporativas utilizan Oracle por su capacidad de manejar operaciones críticas y requerimientos complejos de negocio [27]. Es un sistema propietario, con coste de licencia, pero con soporte empresarial de primer nivel.

- **Microsoft SQL Server** – Sistema de base de datos relacional desarrollado por Microsoft, muy difundido especialmente en entornos Windows y empresariales. SQL Server se integra de forma nativa con todo el **ecosistema Microsoft** (por ejemplo, herramientas como Excel, .NET, Power BI) facilitando soluciones de inteligencia de negocios end-to-end [28]. Es conocido por su facilidad de administración mediante interfaces gráficas (SQL Server Management Studio), su soporte técnico profesional y características como servicios de reporting, análisis y Data Warehousing incorporados. Es una solución comercial orientada a empresas que buscan rendimiento sólido y una integración estrecha con productos Microsoft.

- **SQLite** – A diferencia de las anteriores, SQLite es una biblioteca *embebida* que implementa un motor de base de datos relacional **ligero y minimalista**. No requiere un proceso servidor separado, ya que almacena la base de datos en un simple archivo y las aplicaciones lo acceden directamente. Por su **pequeño tamaño y cero configuración**, SQLite es ideal para aplicaciones móviles, dispositivos o proyectos pequeños que necesitan almacenamiento local [29]. Por ejemplo, muchos aplicativos en Android o iOS usan SQLite internamente para gestionar sus

datos. Aunque no está pensado para grandes cargas concurrentes, es extremadamente eficiente en entornos con recursos limitados y es de dominio público (sin licencia propietaria).

Cabe mencionar que, además de estas opciones relacionales, en la última década han surgido las **bases de datos NoSQL** (no relacionales) como alternativa para ciertos casos de uso. Las bases NoSQL (por ejemplo MongoDB, Cassandra, Redis) se enfocan en esquemas flexibles o datos no estructurados, alta escalabilidad horizontal y almacenamientos especializados (documentos, grafos, etc.). Si bien estas tecnologías han ganado popularidad para *big data* y aplicaciones en tiempo real, las **bases de datos relacionales SQL continúan siendo las más utilizadas en la industria** hoy en día, y a menudo ambas coexisten como soluciones complementarias [30]. La elección depende de las necesidades: las relacionales brindan estructura y consistencia ideales para datos altamente estructurados y transaccionales, mientras que las NoSQL ofrecen flexibilidad y rendimiento en escenarios específicos.

Herramientas para consultar una base de datos

Para explotar y manejar la información almacenada en una base de datos relacional se emplean principalmente **consultas SQL** ejecutadas a través de diversas herramientas. El **lenguaje SQL** (Structured Query Language) es la forma estándar de interactuar con la base de datos – a través de comandos SQL se pueden solicitar datos, insertarlos, actualizarlos o borrarlos [16]. Estas consultas pueden realizarse de dos formas básicas:

1. **Mediante una interfaz de línea de comandos (CLI)**: Todos los SGBD incluyen un programa cliente de consola para enviar comandos SQL directamente. Por ejemplo, MySQL/MariaDB ofrece el cliente `mysql` en consola, PostgreSQL ofrece `psql`, Oracle tiene `SQL*Plus`, SQL Server cuenta con `sqlcmd`, etc. A través de estas consolas, un administrador o desarrollador puede conectarse al servidor de base de datos y escribir instrucciones SQL manualmente para definir esquemas o consultar/modificar datos.

2. **Mediante herramientas gráficas o IDE**: Existen numerosas aplicaciones con interfaz visual que facilitan la administración y consulta de bases de datos. Algunas son provistas por los mismos fabricantes de SGBD y otras son herramientas de terceros. Por ejemplo: **MySQL Workbench** (de Oracle/MySQL) es una GUI para diseñar y consultar MySQL; **pgAdmin** es la herramienta visual estándar para PostgreSQL; **SQL Server Management Studio (SSMS)** para SQL Server; **Oracle SQL Developer** para Oracle; y también herramientas multipropósito como **DBeaver, TablePlus, Navicat**, entre otras, que permiten conectarse a distintos motores. Estas interfaces gráficas suelen ofrecer navegadores de objetos (para ver tablas, vistas, etc.), editores SQL con ayudas sintácticas y visores de resultados.

Usando cualquiera de estas herramientas, el proceso típico para **consultar una base de datos** implica **establecer una conexión** con el servidor y luego ejecutar las consultas SQL deseadas. Muchas herramientas visuales permiten crear una nueva conexión simplemente ingresando los datos de autenticación (usuario y contraseña) y de ubicación del servidor, tras lo cual muestran de inmediato la lista de tablas y esquemas disponibles para interactuar [31]. Por ejemplo, en un cliente como Navicat o MySQL Workbench, el usuario crea una "conexión" indicando el host (servidor), el puerto de red (si es aplicable), el nombre de la base de datos a usar, y las credenciales de acceso; una vez conectado, puede explorar la estructura de la base de datos, construir consultas mediante editores gráficos, editar registros manualmente o importar/exportar datos.

En entornos de desarrollo de software, también es común integrar las consultas desde el código de la aplicación. Para ello se utilizan *librerías* o *drivers* específicos (por ejemplo, controladores ODBC/JDBC,

frameworks ORM, etc.) que permiten al programa enviar comandos SQL al SGBD y recuperar resultados. No obstante, incluso en estos casos subyacentes el mecanismo sigue siendo ejecutar consultas SQL contra la base de datos. En resumen, **SQL es la herramienta central para consultar datos**, ya sea directamente por un usuario técnico mediante consola o GUI, o indirectamente a través de una aplicación. La amplia disponibilidad de herramientas de consulta, tanto CLI como visuales, ha hecho que trabajar con bases de datos relacionales sea accesible y eficiente para administradores, analistas y desarrolladores por igual.

Instalación de la base de datos y sus herramientas utilitarias

Para poner en marcha una base de datos relacional en un entorno local o en un servidor, es necesario **instalar el SGBD** seleccionado junto con las herramientas asociadas. A continuación se describen los pasos generales para la instalación de un sistema de base de datos y sus utilidades (pueden variar en detalle según el producto, pero en esencia son similares):

1. **Descarga e instalación del SGBD**: Obtener el instalador o paquete del sistema de base de datos elegido (por ejemplo, MySQL, PostgreSQL, Oracle, SQL Server, etc.) desde el sitio oficial o repositorio, y ejecutarlo en el sistema operativo de destino. En sistemas Linux, esto podría hacerse vía gestor de paquetes; en Windows o macOS típicamente mediante un instalador gráfico.

2. **Selección de componentes**: Durante el proceso de instalación, normalmente se ofrece elegir qué componentes instalar. Por ejemplo, en MySQL el instalador permite optar por una instalación *completa* que incluye el servidor, herramientas como MySQL Shell, MySQL Workbench, conectores, ejemplos, etc., o bien una instalación *personalizada* donde el usuario marca específicamente los componentes deseados [32]. Para un entorno de desarrollo suele elegirse la opción completa que instala tanto el motor de base de datos como las utilidades gráficas y de línea de comando necesarias [33].

3. **Configuración inicial**: Tras copiar los archivos, se configuran parámetros básicos de la instancia de base de datos. Esto incluye definir el **tipo de despliegue** (por ejemplo, servidor standalone, clúster, desarrollo, etc.), establecer el **puerto de escucha** del servicio (el valor por defecto suele ser 3306 para MySQL/MariaDB, 5432 para PostgreSQL, 1433 para SQL Server, etc.) [34], y **crear las credenciales del administrador**. En muchos SGBD el usuario administrador predeterminado se llama `root` (o `postgres` en PostgreSQL), al cual se le asigna una contraseña durante la instalación [35]. Algunos instaladores permiten también crear usuarios adicionales o ajustar opciones de seguridad (por ejemplo, configurar el servicio de Windows, como en MySQL, o definir políticas de autenticación). Esta configuración inicial asegura que el servidor arranque con los parámetros correctos y protegido con una contraseña administrativa.

4. **Finalización e inicio del servicio**: Completado el asistente o script de instalación, el instalador normalmente inicia el servicio de base de datos para verificar que todo funcione. Se muestran mensajes de éxito y se puede realizar una conexión de prueba para asegurar que la base de datos responde. Es importante anotar datos como la contraseña del administrador y el puerto configurado, para usarlos posteriormente. Tras la instalación, el motor de la base de datos queda ejecutándose (como servicio del sistema) a la espera de conexiones de clientes.

5. **Instalación de herramientas cliente**: En caso de que las herramientas utilitarias de administración no se hayan incluido en el paso 2 (por ejemplo, algunos instaladores de PostgreSQL no traen pgAdmin, o en SQL Server el **Management Studio** se instala por

separado), se debe proceder a instalar estas herramientas manualmente. Esto puede implicar descargar e instalar la consola o entorno gráfico correspondiente. Por ejemplo, para administrar MySQL se puede instalar **MySQL Workbench** (si no se seleccionó antes), para Oracle se puede instalar **SQL Developer** o utilizar herramientas web como Oracle APEX, etc. Estas utilidades no son estrictamente requeridas para que el servidor de base de datos funcione, pero **facilitan enormemente las tareas de administración y consulta**, por lo que forman parte esencial del entorno de trabajo con bases de datos.

En entornos de desarrollo locales es común simplificar este proceso usando paquetes pre-configurados como **XAMPP/WAMP/MAMP** (que incluyen MySQL/MariaDB junto con servidores web y PHP) o contenedores **Docker** con imágenes de la base de datos ya listas para usar. Sin embargo, comprender la instalación manual como se detalló arriba resulta educativo y necesario para configuraciones de producción. Una vez instalado el SGBD y sus herramientas, se estará listo para crear bases de datos, definir las estructuras de tablas y comenzar a almacenar y gestionar información de la organización.

Creación de una conexión a la base de datos

Una vez que el servidor de base de datos está en funcionamiento, el siguiente paso para interactuar con él es **establecer una conexión** desde un cliente o aplicación. Para ello se requieren ciertos **parámetros básicos de conexión** que identifican dónde está la base de datos y con qué credenciales acceder. En general, se necesitan **cuatro datos esenciales**: el *servidor* (nombre de host o dirección IP donde reside la base de datos), el *nombre de la base de datos* específica a la que se desea acceder, el *nombre de usuario* con permisos en dicha base, y la *contraseña* correspondiente a ese usuario [36]. Con esta información, el cliente podrá localizar el servicio de base de datos y autenticarse correctamente en él.

Por ejemplo, si la aplicación cliente se ejecuta en el mismo equipo que la base de datos, el **host** suele ser `"localhost"` (indicando la máquina local) [37]. En caso contrario, se debe proporcionar la dirección de red o IP del servidor donde corre el SGBD (por ej., un dominio o IP pública si es un servidor remoto). Junto con el host, normalmente se especifica el **número de puerto** en el que escucha la base de datos (si se usa el puerto por defecto de ese SGBD, a veces las herramientas lo rellenan automáticamente; por ejemplo, 3306 para MySQL, 1521 para Oracle, etc.). También se indica el **nombre de la base de datos** o esquema al que queremos conectarnos (algunos sistemas permiten omitirlo inicialmente y elegirlo luego, pero es habitual especificarlo para ir directo a los datos deseados).

Las **herramientas gráficas de administración** ofrecen típicamente un diálogo de *nueva conexión* donde se rellenan estos campos. Por ejemplo, en MySQL Workbench al crear una conexión se pide: nombre identificativo para la conexión, hostname (servidor o endpoint), puerto, usuario y opción para guardar la contraseña [38] [39]. En pgAdmin, Navicat u otras, de forma similar se ingresan host, puerto, usuario, password y base de datos a usar. Una vez ingresados, se puede probar la conexión con un botón dedicado (**"Test Connection"**) que verifica si los parámetros son correctos y el servidor responde [40]. Si la prueba es exitosa, la conexión queda almacenada en la herramienta y el usuario puede conectar y desconectar fácilmente en el futuro sin reingresar datos.

En el caso de establecer la conexión desde una **aplicación de software**, estos mismos parámetros se suelen combinar en una *cadena de conexión* (connection string) cuyo formato depende del driver utilizado. Por ejemplo, una cadena de conexión JDBC para MySQL puede verse así: `jdbc:mysql://<servidor>:3306/<basedatos>?user=<usuario>&password=<contraseña>`. Al ejecutar el programa, esta cadena le indica al driver cómo conectarse al SGBD. De forma análoga, lenguajes como

PHP utilizan funciones (mysqli, PDO) donde se pasan servidor, usuario, contraseña y base de datos por separado para conectar [36].

Una vez establecida la conexión, el cliente ya puede enviar consultas SQL al servidor y recibir los resultados. Es importante asegurarse de cerrar la conexión cuando ya no se necesite (en herramientas GUI esto ocurre al salir o desconectar, en aplicaciones de software mediante comandos de cierre), para liberar recursos tanto del lado cliente como del lado servidor. Adicionalmente, en entornos de producción se deben usar conexiones seguras si el tráfico pasa por redes no confiables — por ejemplo habilitando SSL/TLS en la conexión, lo cual muchos SGBD soportan y las herramientas permiten configurar en opciones avanzadas.

En resumen, crear una conexión a la base de datos implica proporcionar las **credenciales y la dirección del servicio** de base de datos al cliente. Con ello se abre un canal de comunicación a través del cual se intercambian comandos (consultas) y datos. Este es el paso previo indispensable para cualquier operación: sin una conexión activa, no es posible consultar ni modificar la información en el SGBD. Con la conexión correctamente configurada y probada, el usuario o la aplicación ya pueden empezar a interactuar con la base de datos de forma segura y eficiente, aprovechando todo el poder de gestión de información que brinda el sistema. [36] [37]

---

[1] [2] [8] [9] [10] Importancia de una base de datos en las empresas en la actualidad – QLU
https://qlu.ac.pa/importancia-base-datos-empresas-actualidad/

[3] [4] [5] [6] [7] [30] Por Qué las Bases de Datos Son Cruciales para las Empresas
https://www.datacentric.es/blog/bases-datos/importancia-bases-de-datos-2/

[11] [12] [19] [20] [23] ¿Qué es el SGBD (sistema de gestión de bases de datos)? | OVHcloud Global
https://www.ovhcloud.com/es/learn/what-is-dbms/

[13] [14] [15] [16] [21] [22] [24] [25] [26] [27] [28] [29] ¿Qué son las Bases de Datos relacionales?: Tipos y características clave
https://www.tdscode.dev/blog/que-son-las-bases-de-datos-relacionales-tipos-y-caracteristicas-clave

[17] [18] ¿Qué es la base de datos ACID? | Pure Storage
https://www.purestorage.com/es/knowledge/what-is-database-acid.html

[31] 5 Programas útiles para gestionar Bases de Datos
https://nubecolectiva.com/blog/5-programas-utiles-para-gestionar-bases-de-datos/

[32] [33] [34] [35] Guía de Instalación de MySQL Server y MySQL Workbench en Diferentes Sistemas Operativos" | Alura Cursos Online
https://www.aluracursos.com/blog/descargar-mysql-serve

[36] [37] Tutorial para Conectar PHP con Bases de Datos MySQL
https://www.hostinger.com/mx/tutoriales/conectar-php-mysql

[38] [39] [40] Conexión desde MySQL Workbench - Amazon Relational Database Service
https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/USER_ConnectToInstance.MySQLWorkbench.html