

# BASES DE DATOS



**SONIA ORIVE**

*Bases de datos relacional*



# BASES DE DATOS (BBDD)

Las bases de datos son sistemas estructurados de forma lógica para la administración electrónica de datos que, con ayuda de un sistema de gestión de bases de datos (**database management system, DBMS**), regulan las pertenencias y los derechos de acceso y guardan la información, añadiéndola al repositorio que contienen. La mayoría de bases de datos solo pueden abrirse, editarse y consultarse con aplicaciones específicas. En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

Una base de datos (**database**) **almacena** datos y los conecta en una **unidad lógica** junto a los metadatos necesarios para su procesamiento. Las bases de datos son instrumentos de gran utilidad para gestionar grandes ficheros y facilitar la consulta de información. En muchas, además, puede definirse un esquema de permisos que establece qué personas o programas pueden acceder a los datos, y a cuáles, con el objetivo de presentar el contenido de forma adecuada y clara.

Los distintos **sistemas de bases de datos** se diferencian conceptualmente entre sí y tienen, por lo tanto, sus propias ventajas y desventajas. Pero, antes que nada, es conveniente diferenciar entre la base de datos en sí y el sistema que la gestiona. Como base de datos se designa al conjunto de los datos que se ha de ordenar, mientras que el **sistema de gestión de la base de datos (SGBD)** es responsable de su administración, determinando así su estructura, el orden, los permisos de acceso, las dependencias, etc.

Para ello acostumbra a utilizar un compilador propio y un modelo adecuado de base de datos que determina la arquitectura del sistema de base de datos.

En muchos casos, solo ciertas aplicaciones, o aquellas que han sido exactamente definidas para ello, pueden leer estos sistemas. Es aquí donde, con frecuencia, se dan confusiones terminológicas cuando un **programa de base de datos** se define solo como "base de datos". El término, además, se utiliza para referirse a simples colecciones de archivos, mientras que, en su sentido estricto, una carpeta con archivos en un ordenador no constituye una base de datos.

Hoy, los sistemas de bases de datos son **imprescindibles** en numerosos campos. Cualquier tipo de software concebido para las empresas se basa en robustas bases de datos con un gran número de opciones y herramientas para los administradores del sistema. La **seguridad de los datos**, además, ha ido ganando importancia con el tiempo, y es que en las bases de datos electrónicas se almacenan y cifran **contraseñas, datos personales** e incluso **divisa** digital.

El sistema financiero moderno, no es más que una red de bases de datos, en la cual la mayor parte de las cuantías monetarias solo existen como unidades electrónicas de información, cuya protección, por medio de bases de datos seguras es una de las tareas principales de las instituciones financieras. Aunque no solo por esto son cruciales las bases de datos electrónicas para la civilización moderna.



# SISTEMA DE GESTIÓN DE BASES DE DATOS(SGBD)

## Funciones y condiciones

Un término muy extendido para describir las funciones y los requisitos de las transacciones en un database management system es el de ACID, acrónimo de atomicity, consistency, isolation y durability (atomicidad, consistencia, aislamiento, durabilidad). Estos cuatro parámetros, cubren los requisitos más importantes de un SGBD (ACID compliant):

- **Atomicidad** designa a la propiedad “todo o nada” de los gestores de bases de datos: para que una consulta sea válida y la transacción se complete correctamente se ha de llevar a cabo en el orden correcto de pasos.
- **La consistencia** (o coherencia) se da cuando al finalizar una transacción, la base de datos sigue siendo estable, lo que requiere la supervisión continua de todas las transacciones.
- **El aislamiento** es la condición que garantiza que las transacciones no se obstaculicen unas a otras, algo que normalmente se logra con ciertas funciones de bloqueo que aíslan los datos que participan en una transacción.
- **La durabilidad** significa que en un SGBD todos los datos se guardan a largo plazo incluso tras concluir una transacción y también, o especialmente, en el caso de fallos del sistema o caídas del SGBD. Para esta condición, son esenciales los registros de transacción, que protocolizan todos los procesos que tienen lugar en el SGBD.





# Tabla de funciones de un sistema de gestión

Función/condición	Significado
Almacenar datos	Las bases de datos almacenan textos, documentos, contraseñas, etc., en formato electrónico, a los que puede accederse mediante consultas.
Editar datos	Según de qué permisos se disponga, la mayoría de bases de datos permiten editar <i>in situ</i> los datos que salvaguardan.
Borrar datos	Los registros de las bases de datos pueden borrarse por completo, sin dejar espacios en blanco. En algunos casos los datos que se han borrado pueden restablecerse, pero en otros, se eliminan definitivamente.
Gestionar los metadatos	Normalmente, la información se guarda con metadatos o metaetiquetas que mantienen el orden dentro de la base de datos y hacen posible la función de búsqueda. Los metadatos también suelen utilizarse para regular los permisos.  La gestión de datos comprende cuatro operaciones fundamentales: crear (create), leer/recuperar (read/retrieve), actualizar (update) y borrar (delete). Este concepto, conocido por su acrónimo <a href="#">CRUD</a> , constituye la base de la gestión de datos.
Seguridad de los datos	Las bases de datos han de ser seguras para evitar que sujetos no autorizados puedan acceder a la información que guardan. Además de un solvente método de cifrado, para mantener la seguridad de los datos es esencial poner esmero en su administración, sobre todo su administrador principal. La seguridad de los datos implica tomar las precauciones técnicas necesarias para impedir la manipulación o la pérdida de datos.
Integridad de los datos	La integridad de los datos significa que los datos han de cumplir con ciertas reglas para asegurar su corrección y definir la lógica de negocio del banco de datos. Solo así, puede asegurarse que la base de datos ,al completo, funciona de forma constante y coherente. En los modelos relacionales se dan cuatro de estas reglas: integridad de campo, integridad de entidad, integridad referencial y consistencia lógica.
Función multiusuario	Las aplicaciones de base de datos permiten acceder a las bases de datos desde diferentes dispositivos. El reparto de permisos y la seguridad de los datos son elementales en el uso multiusuario. También constituye un reto, mantener la consistencia de los datos sin dificultar el rendimiento, cuando varios usuarios leen y escriben a la vez.
Optimizar las consultas	Técnicamente, una base de datos ha de poder procesar las consultas de la mejor manera posible para garantizar una buena <i>performance</i> . Si utiliza demasiadas rutas diferentes para solucionar una consulta, el rendimiento global del sistema se verá perjudicado.
Triggers y stored procedures	Estos dos procedimientos son minipliegues guardados en los SGBD que se activan con ciertos eventos. Con ellos se pretende, entre otras cosas, mejorar la integridad de los datos. Los disparadores (triggers) y los procedimientos almacenados (stored procedures) son procesos típicos de las bases de datos relacionales. Los segundos contribuyen a la seguridad del sistema si los usuarios solo ejecutan las acciones con procedimientos predefinidos.
Transparencia del sistema	La transparencia del sistema es relevante, sobre todo, en los sistemas distribuidos; privando al usuario de la distribución y la implementación de los datos, la utilización de una base de datos distribuida se asemeja al de una centralizada. Los procesos que corren en segundo plano se muestran u ocultan en diversos niveles de transparencia. La función principal es, no obstante, simplificar su uso todo lo posible.



Hasta hoy la gestión electrónica de datos ha estado dominada por el modelo de base de datos relacional. Entre los gestores de bases de datos relacionales más utilizados se cuentan, por orden alfabético:

## Db2

con Db2 los usuarios disponen de un SGBD relacional propietario de la casa IBM.

## Microsoft SQL Server

la aplicación de Microsoft para gestionar bases de datos relacionales está disponible con una licencia Microsoft de pago

## MySQL

MySQL es el SGBD de código abierto más utilizado a nivel global. Desde que pasa a las manos de Oracle, MySQL se distribuye con una licencia dual. Sus primeros desarrolladores siguen encargándose del proyecto, ahora bajo el nombre de MariaDB.

## PostgreSQL

con PostgreSQL los usuarios disponen de un SGBD relacional libre y orientado a objetos de cuyo continuo desarrollo se ocupa su comunidad open source.

## Oracle Database

el programa de Oracle se distribuye como software propietario.

## SQLite

SQLite constituye una biblioteca de programas con licencia de dominio público que contiene un gestor de bases de datos relacionales.

Todos estos sistemas se basan en una organización tabular de la información.



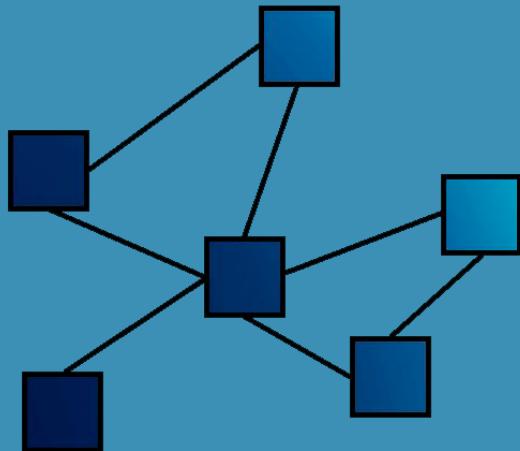
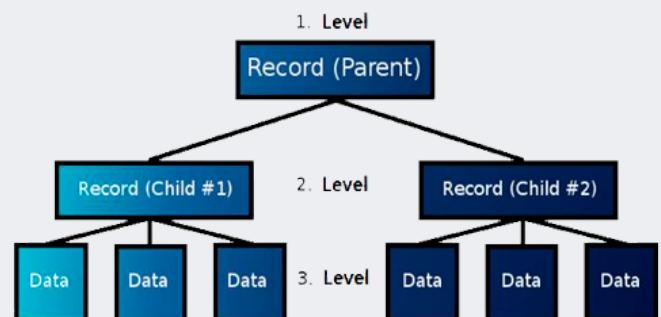
# MODELOS DE BASES DE DATOS

## Evolución de los modelos

Las diferencias entre los modelos de bases de datos más habituales es resultado de la evolución técnica de la transmisión electrónica de datos, que no solo perseguía la eficiencia y la manejabilidad, sino también, el empoderamiento de los fabricantes más renombrados.

### Modelo jerárquico de base de datos

Este es el modelo más antiguo, hoy superado por el modelo relacional (entre otros). XML utiliza este sistema para guardar datos y algunas compañías de seguros y bancos recurren a las bases de datos jerárquicas, sobre todo, en las aplicaciones más antiguas de base de datos. El sistema de base de datos jerárquico más conocido es IMS/DB de IBM.



### MODELO EN RED

El modelo en red se desarrolló casi de forma simultánea al relacional. Cada registro puede tener múltiples precedentes, lo que le da la estructura en red de su nombre. Para acceder a un registro no hay un camino único e invariable. Hoy el modelo de base de datos en red se utiliza, sobre todo, en los grandes ordenadores. Algunos modelos conocidos de base de datos en red son el UDS de Siemens y el DMS de Sperry Univac.



# Modelo orientado a objetos

Estas bases de datos, disponibles en formato open source, , suelen utilizarse en plataformas Java y .NET. La más conocida es db4o, que destaca, sobre todo, por un escaso uso de la memoria. Acostumbran a trabajar con el lenguaje OQL, muy similar a SQL. Los datos se guardan en un **objeto** junto con sus funciones (métodos) y los atributos que los describen más en profundidad. Los métodos, depositados en el objeto junto con los datos, son los que definen cómo se accede al objeto.

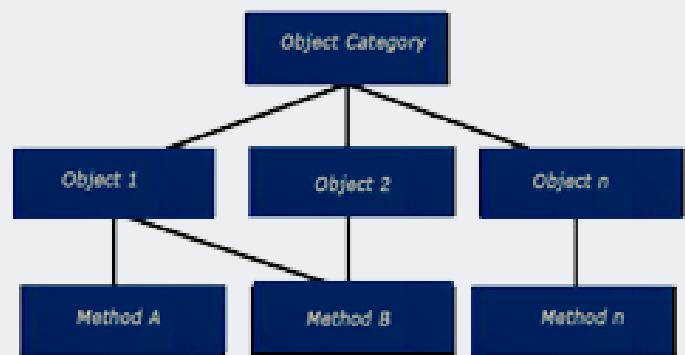
Los objetos pueden ser **complejos** y estar compuestos por múltiples tipos de datos, son únicos dentro del sistema de base de datos y se identifican con un **identificador de objeto** (OID en inglés) único.

Los objetos se agrupan en clases (object category), dando como resultado una jerarquía de clases.

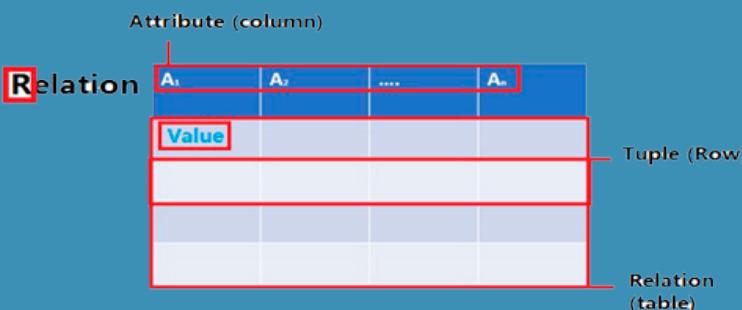
No existe ninguna relación padre-hijo fija, aún así, a través de la clase puede definirse el método para el acceso.

Entre sus ventajas destacan, sobre todo, en problemas con tipos de datos complejos.

Trabajan, en su mayor parte, de forma autónoma sin recurrir a la normalización y a la correspondencia de ID, permitiendo así almacenar los objetos nuevos de forma relativamente simple y fluida. Sin embargo, las consultas son mucho más ágiles en un sistema de base de datos relacional. La escasa popularidad de los sistemas orientados a objetos resulta en una insuficiente compatibilidad con muchas de las aplicaciones de base de datos que se usan habitualmente.



## MODELO RELACIONAL



El modelo de más popularidad a día de hoy es el relacional, su sistema de gestión es más conocido como SGBDR (RDBMS en inglés) y como lenguaje utiliza normalmente SQL. **Basado en tablas**, gira en torno al concepto de relación, un término que aquí se utiliza como sinónimo de tabla. Para formular las relaciones se utiliza álgebra relacional, con cuya ayuda puede obtenerse la información de estas relaciones. Este es el principio que fundamenta el lenguaje SQL.

El modelo relacional trabaja con **tablas** independientes que determinan la localización de los datos y sus conexiones. Estos datos conforman un registro (en la imagen, una fila o "tupla") y se guardan en columnas como atributos (en la imagen, de A1 a An). La **relación** es lo que resulta de los atributos interrelacionados. Para identificar inequívocamente un registro es elemental la **clave primaria**, que normalmente se define como el primer atributo (A1) y que no puede cambiarse. Dicho de otra manera, esta clave primaria o ID, define la posición exacta del registro con todos los atributos.



# Modelo orientado a documentos

En este modelo, los **documentos** son la unidad básica para el almacenamiento de datos. Estas unidades son las que estructuran los datos y no deben confundirse con los documentos de los programas de procesamiento de texto. Aquí, los datos se guardan en los llamados **pares clave-valor**, comprendiendo así, una “clave” y un “valor”.

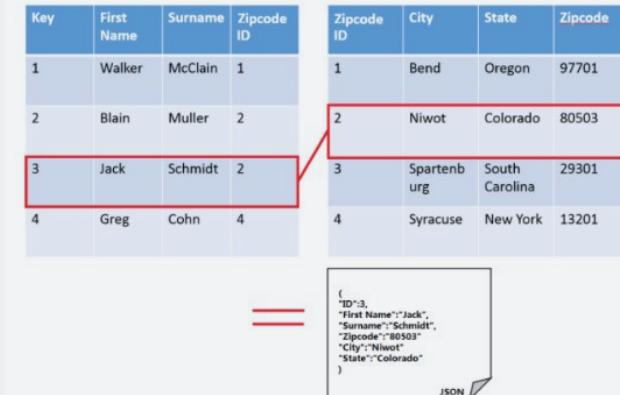
Como no están definidos ni la estructura ni el número de pares, los documentos que integran una base de datos orientada a documentos pueden resultar muy dispares entre sí.

Cada documento es una unidad cerrada en sí misma y establecer relaciones entre documentos no resulta fácil, pero en este modelo no es necesario.

En el modelo relacional (arriba representado con las dos tablas), varias relaciones (tablas) se conectan entre sí para seleccionar un registro común. En el modelo documental, **un único documento basta para guardar toda la información**. Aquí no se está obligado a utilizar un determinado esquema porque, mientras se use siempre el mismo lenguaje de base de datos, este modelo está conceptualmente **libre de esquemas**.

Una idea fundamental de las bases de datos documentales es que los datos que guardan relación entre sí siempre **se guardan juntos en un lugar** (en el documento). Mientras que las bases de datos relacionales suelen representar y mostrar la información relacionada conectando varias tablas, en el modelo que nos ocupa es suficiente con consultar un solo documento. Esto reduce el número de procedimientos necesarios para consultar la base de datos.

Estos sistemas son especialmente interesantes para las aplicaciones web, puesto que permiten guardar formularios HTML completos. Fue sobre todo con el avance de la web 2.0 cuando estas bases de datos vieron aumentar su popularidad. Con todo, es necesario remarcar que entre los diversos sistemas basados en documentos se dan diferencias notables, desde la sintaxis hasta la estructura interna, por lo que no todas las bases de datos orientadas a documentos son apropiadas para cualquier escenario. Es debido a estas diferencias por lo que hoy disponemos de algunos sistemas de bases de datos orientados a documentos de la reputación de Lotus Notes, Amazon SimpleDB, MongoDB, CouchDB, Riak, ThruDB, OrientDB, etc.





# Modelos de bases de datos y características

Modelo BBDD	Desarrollo	Ventajas	Inconvenientes	Ámbitos de aplicación	Marcas
<b>Jerárquico</b>	Década de 1960	Acceso de lectura muy rápido, estructura clara, técnicamente simple	Estructura fija en árbol que no permite conexiones entre árboles	Bancos, compañías de seguros, sistemas operativos	IMS/DB
<b>En red</b>	Principios de la década de 1970	Admite varias formas de acceder a un registro, sin jerarquía estricta	En bases de datos más grandes no se tiene una vista general	Grandes ordenadores	UDS (Siemens), DMS (Sperry Univac)
<b>Relacional</b>	1970	Simple, flexible. Creación y edición fácil y flexible, fácil de ampliar, rápida puesta en marcha, contexto de competencia muy dinámica	Inmanejable con cantidades grandes de datos, segmentación deficiente, atributos de clave artificiales, interfaz de programación externa, no refleja bien las propiedades y la conducta de los objetos	Control de gestión (controlling), facturación, sistemas de control de inventario, sistemas de gestión de contenido, etc.	MySQL, PostgreSQL, Oracle, SQLite, DB2, Ingres, MariaDB, Microsoft Access
<b>Orientado a objetos</b>	Finales de la década de 1980	Soporta mejor los lenguajes de programación orientados a objetos. Permite almacenar contenido multimedia	El rendimiento empeora con grandes volúmenes de datos, pocas interfaces compatibles	Inventario (museos, comercio minorista)	db4o
<b>Orientado a documentos</b>	Década de 1980	Los datos relacionados se guardan de forma centralizada en documentos independientes, estructura libre, concepción multimedia	El trabajo de organización es relativamente alto, a menudo requiere conocimientos de programación	Aplicaciones web, buscadores, bases de datos de texto	Lotus Notes, Amazon SimpleDB, MongoDB, CouchDB, Riak, ThruDB, OrientDB



# BASES DE DATOS RELACIONALES

## ¿Qué es una base de datos relacional ?

Un concepto capital del modelo relacional es el de relación, postulado por el matemático y teórico de bases de datos Edgar F. Codd. Siguiendo al científico británico, una relación representa un conjunto de entidades con las mismas propiedades. Cada relación se compone de una serie de filas o registros (las llamadas tuplas), cuyos valores dependen de ciertos atributos (columnas).

Para definir los atributos de una relación y el tipo de dato (dominio) permitido para estos valores, se utiliza un esquema con esta sintaxis:

$$R = (A_1:D_1, A_2:D_2, \dots, A_n:D_n)$$

Aquí, la relación R comprende de los atributos A1 a An y cada atributo corresponde a un tipo de dato o dominio (D1, D2 , etc.).

```
1 Empleados = ( e_ID : integer,
2     1er apellido : string
3     2º apellido : string,
4     nombre : string,
5     nº SS : string,
6     calle : string,
7     CP : string,
8     municipio : string )
```

segundo apellido, nombre, número de la seguridad social (nº SS), calle, código postal y municipio, podría utilizarse para la gestión interna de los datos personales de la plantilla de una empresa.

A cada atributo le corresponde un tipo de dato, string o integer, lo que indica que hay atributos que esperan cómo valor una secuencia de caracteres (string) y otros que solo aceptan números enteros (integer).

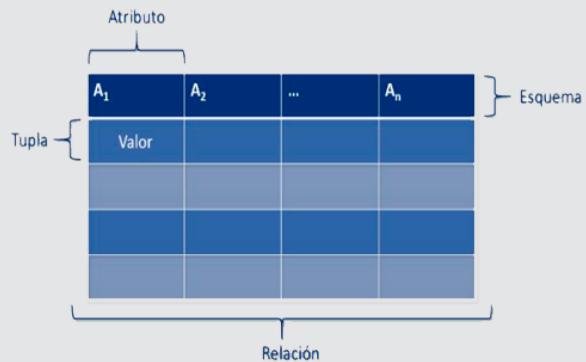
Una relación con el esquema explicado arriba podría contener la siguiente tupla (fila):

```
1 | (1, García, Fernández, Antonio, 32 12345678 12, Calle Principal 1, 11111, Villarriba)
```





La tabla, concepto clásico de la organización de la información, es el formato que utiliza el modelo relacional para explicar de un modo visual la ordenación de los valores de una tupla en función de los atributos definidos en la relación. Una base de datos relacional no es otra cosa, entonces, que un conjunto de tablas interrelacionadas.

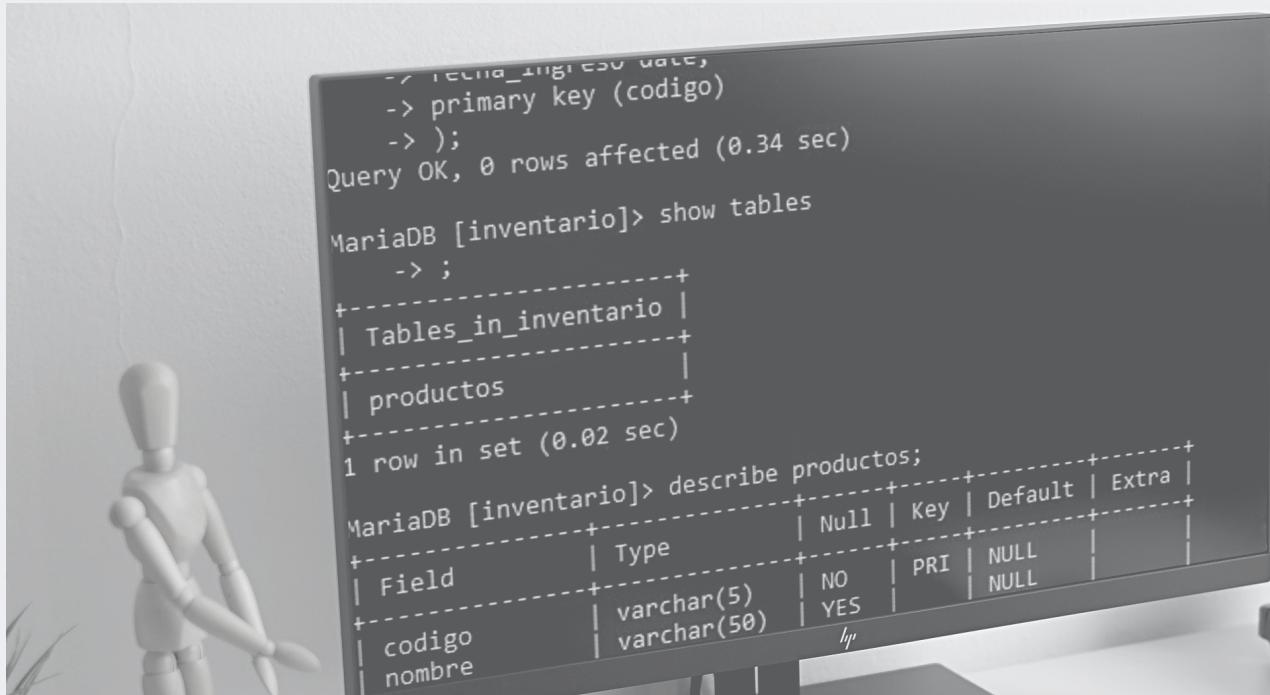


Las tablas son sistemas de clasificación constituidos por filas horizontales y columnas verticales que permiten agrupar datos y presentarlos de forma ordenada. Cada fila de una tabla se denomina tupla. Los valores que contiene cada tupla vienen determinados por los atributos definidos en el esquema relacional.

Ejemplo de una tabla tal como resultaría del esquema anterior:

	e_ID	1er apellido	2º apellido	nombre	nº SS	calle	CP	municipio	coche_ID
1	García	Fernández	Antonio	12	32 12345678 12	Calle Principal 1	11111	Villarriba	3
2	García	García	Josefa	49	28 87654321 49	Calle Iglesia 2	22222	Villabajo	1
3	Expósito	Hernández	Gonzalo	3	25 091225 46 3	Plaza Mercado 3	33333	Campoarriba	1
4	Casas	González	Antonia	78	23 170839 78 78	Calle Grande 4	44444	Campoabajo	2

Esta tabla guarda los datos de la plantilla de una empresa y se compone de cuatro registros, cada uno de los cuales contiene información sobre un solo empleado.





# ¿Cómo funcionan las bbdd relacionales?

Los datos estructurados en tablas constituyen la BD de un sistema relacional. El SGBD define su estructura y gestiona también los permisos de escritura y lectura y para interactuar con él, los usuarios utilizan un **lenguaje de bases de datos**.

Todo gestor de bases de datos relacionales soporta al menos un lenguaje formal que permite ejecutar las siguientes operaciones:

- **Definir la estructura de datos:** en la definición de los datos se guarda una descripción con metadatos de la estructura de datos en el diccionario del sistema. Cuando un usuario crea una tabla nueva, en el diccionario de datos se almacena su correspondiente esquema. El vocabulario de un lenguaje de bases de datos que se utiliza para definir los datos se denomina *Data Definition Language (DDL)*, lenguaje de definición de datos.
- **Definir condiciones de integridad:** por condiciones de integridad se entienden los requisitos de estado que se exigen a un banco de datos. Si se definen condiciones para su integridad, la BD garantiza que se cumplan en todo momento. Se habla entonces de un estado consistente. Una condición básica de integridad en una base de datos relacional es, por ejemplo, que cada registro (tupla) pueda identificarse de forma inequívoca.
- **Definir transacciones:** cuando se lleva a una BD de un estado consistente a otro diferente se habla de transacción. Estas transacciones contienen una serie de instrucciones que deben ejecutarse siempre de forma íntegra. Si una se interrumpe, la BD vuelve a su estado original (Rollback). Cada transacción comienza con una orden para crear una conexión con la BD a la que siguen otras que inician las operaciones de datos en sí, así como un paso de comprobación (Commit) que asegura la integridad de la BD. Las operaciones que pongan en peligro la integridad de la tabla no se consignan (committed), es decir, no se escriben en la base de datos de forma permanente. Por último, se cierra la conexión con la BD. Al vocabulario del lenguaje de bases de datos con el que se manipulan los datos se le conoce como *Data Manipulation Language (DML)*.
- **Definir vistas:** las llamadas views son vistas virtuales de un subconjunto de los datos de una tabla. Para crear una vista, el SGBD genera una tabla virtual (relación lógica) sobre la base de las tablas físicas. En estas vistas pueden emplearse las mismas operaciones que se utilizarían en tablas físicas. Según la función de la vista de datos pueden distinguirse distintos tipos de vista. Las más habituales son aquellas que filtran determinadas filas (consulta de selección) o columnas (vista de columnas) de una tabla, así como las que conectan diversas tablas entre sí (vista de conjunto).

En el modelo relacional se utiliza de forma estándar para estas operaciones el **lenguaje de bases de datos SQL** (Structured Query Language), basado en el álgebra relacional.



# SQL

Las operaciones típicas de las BD como consultar, crear, actualizar o borrar datos se realizan por medio de las llamadas **sentencias SQL** (SQL statements), una combinación de órdenes SQL, semánticamente vinculadas al inglés y por este motivo bastante elocuentes.

Esta tabla contiene términos fundamentales del modelo de datos relacional y su equivalente en terminología SQL.

Modelo de datos relacional	SQL
Relación	Table (tabla)
Atributo	Column (columna)
Tupla	Row (fila)

```
1 | SELECT columna FROM tabla WHERE columna = valor;
```

En la tabla “Empleados”, esta sentencia SQL podría resultar así.

Esta declaración indica al gestor de la BD que invoque un valor de la columna nº SS de la tabla “Empleados”. La condición que hemos definido es que se seleccione el valor del registro cuyo valor de atributo de la columna e\_ID equivalga al valor 3.

Como resultado, la base de datos entrega el valor 25 091225 46, el número de la seguridad social de Gonzalo Expósito Hernández, el empleado identificado con el número 3.

Para ejecutar una consulta sencilla con SQL podemos utilizar este esquema.

Utilizamos **SELECT** en primer lugar para ordenar al SGBD relacional que consulte ciertos datos. A continuación definimos los datos que queremos consultar proporcionando tanto la tabla como la columna que deseamos seleccionar. Con **WHERE** integraremos una condición en la sentencia SQL porque no queremos abrir todos los valores en esta columna, sino solo el valor de un determinado registro.

```
1 | SELECT nº_SS FROM Empleados WHERE e_ID = 3;
```

## Normalización

Cuando se trabaja con bases de datos relacionales, rara vez se hace con una única tabla. Normalmente se manejan arquitecturas en las cuales los datos se clasifican en tablas separadas en función de su significado. La necesidad de hacer consultas cruzadas para obtener datos guardados en tablas diferentes es la que da origen al concepto que sustenta el modelo relacional de bases de datos.

En principio, la información de una base de datos relacional podría guardarse en una sola tabla, con la ventaja de que no sería necesario interconectar diversas tablas, ni utilizar la compleja sintaxis derivada de las consultas a varias tablas diferentes. Sin embargo, el reparto de información en varias tablas contribuye a reducir las entradas dobles (las llamadas anomalías), un proceso que se conoce como “normalización”.

El grado de normalización de una tabla puede definirse a partir de varias **formas normales**:

- Primera forma normal (1FN)
- Segunda forma normal (2FN)
- Tercera forma normal (3NF)
- Forma normal de BoyceCodd (FNBC)
- Cuarta forma normal (4FN)
- Quinta forma normal (5FN)

En este modelo de datos se llama “relaciones” (relationships) a las relaciones entre tablas separadas por medio de claves. Estas claves son las que conectan las tablas entre sí y permiten que los datos de tablas diferentes puedan consultarse o modificarse siempre con la misma sentencia.



# Formas normales

## PRIMERA FORMA NORMAL (1FN)

Una tabla está en primera forma si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- No debe existir variación en el número de columnas.
- Los campos no clave deben identificarse por la clave (dependencia funcional).
- Debe existir una independencia del orden tanto de las filas como de las columnas; es decir, si los datos cambian de orden no deben cambiar sus significados.

Esta forma normal elimina los valores repetidos dentro de una base de datos.

## SEGUNDA FORMA NORMAL (2FN)

**Dependencia funcional.** Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir, que no existen dependencias parciales. Todos los atributos que no son clave principal deben depender únicamente de la clave principal.

## TERCERA FORMA NORMAL (3NF)

La 3NF fue definida originalmente por E.F. Codd.<sup>2</sup> La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva en los atributos que no son clave

## FORMA NORMAL DE BOYCECODD (FNBC)

La tabla se encuentra en FNBC si cada determinante, atributo que determina completamente a otro, es clave candidata. Deberá registrarse de forma anillada ante la presencia de un intervalo seguido de una formalización perpetua, es decir las variantes creadas, en una tabla no se llegarán a mostrar, si las ya planificadas, dejan de existir.



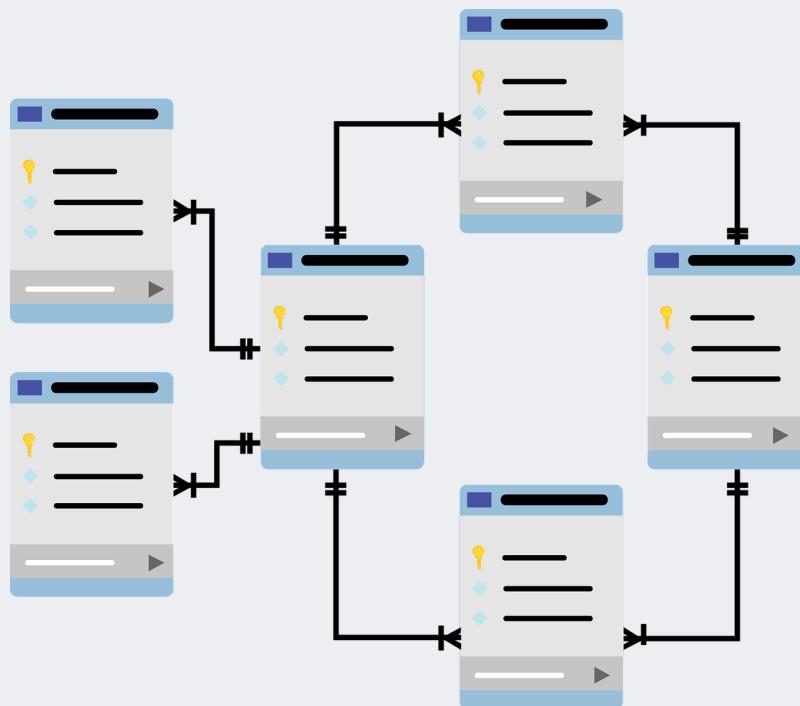
## CUARTA FORMA NORMAL (4FN)

Una tabla está en 4FN si ya pasó por la 3FN o en BCFN y no posee dependencias multivaluadas no triviales. Una tabla con una dependencia multivaluada es una donde la existencia de dos o más relaciones independientes muchos a muchos causa redundancia, la que es suprimida por la cuarta forma normal.

## QUINTA FORMA NORMAL (5FN)

Una tabla se encuentra en 5FN si:

- La tabla está en 4FN
- No existen relaciones de dependencias de reunión (join) no triviales que no se generen desde las claves. Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo si, cada relación de dependencia de reunión (join) se encuentra definida por claves candidatas. Por lo que si se aplicara una consulta entre al menos tres relaciones independientes entre sí dentro de la 4FN y se obtuvieran tuplas espurias, entonces no estaría dentro de la 5FN.





# Claves

Las tablas como la de nuestro ejemplo permiten consultar valores o registros de distintas formas, pero en todas ellas el componente principal son las claves. En el modelo relacional se conoce como claves a los atributos que sirven para identificar un registro de forma inequívoca.

Volviendo a nuestra tabla “Empleados”, la siguiente clave permite identificar una tupla:

1	{e_ID, 1er apellido, 2º apellido, nombre, nº SS}
---	--

Con los datos de la tabla, la siguiente clave serviría para identificar de forma única el registro del empleado Gonzalo Expósito:

1	{e_ID = '3', 1er apellido = 'Expósito', 2º apellido = 'Hernández', nombre = 'Gonzalo', nº SS = '25 091225 46'}
---	--

A estas llaves se las conoce como **superclaves**, si bien en la práctica tienen poca relevancia, entre otras cosas porque a menudo contienen más atributos de los necesarios. En cambio, en el contexto de las bases de datos relacionales, acostumbra a trabajarse con los fragmentos más pequeños de una superclave, los llamados candidatos a clave. Una tabla puede contener varios candidatos a clave que identifican a las tuplas inequívocamente.

Como hemos visto en nuestra consulta anterior, los registros de la tabla “Empleados” pueden identificarse solo con el número de identificación (e\_ID). Otro candidato a clave podría ser el número de la seguridad social (nº SS). Una clave como {apellido, nombre}, en cambio, no sería un buen candidato, ya que esta pareja de atributos no puede asignarse con seguridad a un solo empleado (en una empresa podría encontrarse a varios empleados con el mismo nombre y el mismo primer apellido). En consecuencia, una identificación con esta clave no estaría libre de duda. Sin embargo, no puede haber dos empleados que comparten el ID o número de la seguridad social.

Así, para nuestra tabla pueden definirse los siguientes candidatos a clave:

1	{e_ID}
2	{nº SS}

Las bases de datos relacionales suelen estructurarse de tal forma que uno de los posibles candidatos a clave indique el orden de los registros. Este candidato a clave se convierte entonces en **clave primaria** (primary key) y suele tratarse de números identificativos consecutivos.

En nuestra tabla sería e\_ID.

Por su capacidad para identificar los registros en las bases de datos relacionales, las claves se ajustan a la perfección para interconectar las diferentes tablas que componen una BD. Para hacerlo, la clave primaria de una tabla se convierte en la **clave ajena/foránea** (foreign key) de otra.

Por su capacidad para identificar los registros en las bases de datos relacionales, las claves se ajustan a la perfección para interconectar las diferentes tablas que componen una BD. Para hacerlo, la clave primaria de una tabla se convierte en la clave ajena (foreign key) de otra.



La tabla que presentamos a continuación contiene los datos que una empresa podría haber registrado sobre su parque móvil. La clave primaria de la tabla llamada “Vehículos” es un coche\_ID consecutivo:

coche_ID	marca	modelo	matrícula	fabricación	ITV
1	VW	Caddy	1234 TGB	2016	43452
2	Opel	Astra	9876 ZBU	2010	43689
3	BMW	X6	5847 LOG	2017	43344

Ahora, para mostrar el coche que conduce cada empleado debería conectarse la tabla “Vehículos” con la tabla “Empleados”. Una forma de hacerlo sería integrando la clave primaria de la tabla del automóvil (coche\_ID) como clave ajena en la tabla con los datos sobre la plantilla:

e_ID	1º apellido	2º apellido	nombre	nº SS	calle	CP	municipio	coche_ID
1	García	Fernández	Antonio	32 12345678 12	Calle Principal 1	11111	Villarriba	3
2	García	García	Josefa	28 87654321 49	Calle Iglesia 2	22222	Villabajo	1
3	Expósito	Hernández	Gonzalo	25 091225 46 3	Plaza Mercado	33333	Campoarriba	1
4	Casas	González	Antonia	23 170839 78	Calle Grande 4	44444	Campoabajo	2

Ahora sabemos que el trabajador García Fernández conduce un automóvil con el ID 3, la señora Casas González uno con el ID 2 y que Josefa y Gonzalo comparten el coche con el ID 1.

Si se tratara de saber qué trabajador llevará la próxima vez el coche al taller y cuándo lo hará, se deberá consultar tanto la tabla “Empleados” como la tabla “Vehículos”. Pero como se han conectado mediante la clave ajena, esta consulta puede hacerse una sola vez. Las operaciones de BD que abarcan varias tablas se realizan en el modelo relacional con ayuda de las llamadas sentencias JOIN.

## JOIN

---

JOIN puede traducirse como la acción de “unir” o “combinar” y en este contexto hace referencia a una operación que permite consultar varias tablas de datos simultáneamente. Los datos que se extraen de las tablas seleccionadas se agrupan en un subconjunto con todos los posibles resultados y se entregan en función de las condiciones que se han definido.

Los usuarios deciden con la elección del tipo de JOIN y con ayuda de una condición de selección qué datos se extraen de las tablas.

Entre los **tipos de JOIN** más importantes se cuentan:

- INNER JOIN (combinación interna)
- OUTER JOIN (combinación externa)
- SELF JOIN (una tabla enlaza consigo misma)



# Diferencias con otros modelos de bases de datos

Las bases de datos relacionales basadas en SQL se diferencian de los conceptos que rompen con la estructura fija de las tablas y persiguen un enfoque alternativo a la hora de organizar los datos. Entre los más importantes se encuentran las bases de datos orientadas a objetos y los sistemas basados en documentos.

Con las bases de datos orientadas a objetos (BDOO) entra en escena, a finales de la década de 1980, un nuevo modelo que retoma elementos de la programación orientada a objetos (POO) y permite el almacenamiento de los datos en forma de objetos. Aunque este principio no logra consolidarse, algunas de sus ideas consiguen colarse en el desarrollo de sistemas de bases de datos relacionales. El resultado son productos con extensiones objeto-relacionales que facilitan el almacenamiento de tipos abstractos de datos en el modelo relacional de base de datos.

Con los cambios en el uso de Internet que trajo consigo el nuevo siglo y la web 2.0, el modelo relacional volvió a ser objeto de críticas. En el marco del movimiento NoSQL (Not only SQL), cuyo objeto era crear sistemas de BD de gran rendimiento aptos para aplicaciones con un uso masivo de datos, comienzan a desarrollarse entonces modelos alternativos como el de las bases de datos orientadas a documentos (BDOD). Estos dos modelos, los orientados a objetos y a documentos, se diferencian del modelo relacional sobre todo en la forma como se almacenan y se accede a los datos.

## Bases de datos orientadas a objetos

El modelo orientado a objetos considera el almacenamiento de datos como objetos, los cuales se procesan de forma análoga a como sucede en la **programación orientada a objetos**.

Un objeto define a una entidad y contiene:

- los atributos necesarios para describir a la entidad,
- conexiones (relaciones) con otros objetos,
- funciones que permiten acceder a los datos almacenados (métodos).

Un objeto es, así, una combinación de datos en la cual también se define el punto de acceso a estos datos. Los objetos se consideran **tipos abstractos de datos**.

El SGBD orientado a objetos (ODBMS, object database management system) asigna automáticamente un ID a cada objeto que permite identificarle de forma única y dirigirse a él con métodos.

Este **ID** es independiente del estado del objeto y está desvinculado de sus valores.

Esto permite asignar ID diferentes a dos objetos con los mismos datos, esto es, con un mismo estado. El modelo orientado a objetos se aleja así del relacional, modelo en el que cada tupla puede identificarse a partir de sus datos, por ejemplo, con una clave primaria.

Otra característica del modelo orientado a objetos es el **aislamiento de los datos**: la única forma de acceder a los datos guardados es utilizando los métodos que el usuario ha definido previamente. Esto protege a los datos de cambios procedentes de puntos de acceso no definidos.

Por otro lado, las estructuras de bases de datos se definen aquí por medio de un **sistema de clases jerárquico**. En el contexto de la programación orientada a objetos, una clase es un conjunto de objetos que poseen las mismas propiedades y a cada clase de objetos le subyace una definición de clases.



Este esquema prescribe los atributos y los métodos de todos los objetos de la misma clase, determinando así cómo se crean y se modifican.

Para interactuar con el sistema gestor de estas bases de datos, los usuarios utilizan un lenguaje de consultas inspirado en SQL, el lenguaje de consultas a objetos u OQL (Object Query Language).

Algunas implementaciones conocidas del modelo de BDOO son **Realm**, **ZODB** y **Perst**.

El desarrollo de las BDOO pretendía solucionar un problema de la programación de aplicaciones, la **incompatibilidad de impedancia** objeto-relacional (Object-relational impedance mismatch), que se produce inevitablemente si se guardan objetos de un lenguaje orientado a objetos como C#, C++ o Java en una base de datos relacional y resulta de las diferencias fundamentales entre ambos paradigmas de programación, como son:

- Las bases de datos relacionales no soportan conceptos orientados a objetos como las clases y la herencia.
- El modelo relacional no permite la identificación de objetos independiente del estado.
- El modelo relacional tampoco cuenta con el mecanismo de protección que aporta el aislamiento de datos.

Una forma de evitar estos problemas de incompatibilidad consiste en no utilizar bases de datos relacionales y optar por una BDOO cuando se programan aplicaciones orientadas a objetos.

Sin embargo, la desventaja de esta opción es que los datos encapsulados en objetos dejan de estar disponibles fuera de la aplicación. A esto se añade la **reducida expansión** de este tipo de bases de datos.

La mayoría de herramientas e interfaces que se utilizan para analizar bancos de datos sigue estando diseñada para bases de datos relacionales y no soportan el modelo orientado a objetos.

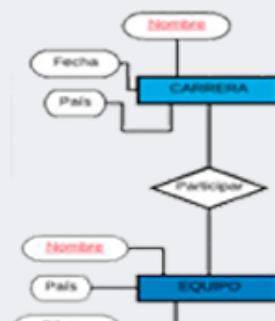
Pese a todo, los programadores de aplicaciones que no quieran prescindir de las ventajas del modelo relacional tienen la posibilidad de compensar estas incompatibilidades con ayuda del mapeo objeto-relacional (O/R mapping o object-relational mapping, ORM).

Las **funcionalidades para la representación objeto-relacional** se implementan con bibliotecas que crean una capa de abstracción entre la aplicación orientada a objetos y los datos guardados en las tablas.

Asimismo, hay muchos fabricantes de sistemas relacionales que equipan sus productos con funciones para compensar estas incompatibilidades con la programación orientada a objetos.

Estos sistemas se conocen como objeto-relacionales.

Las BDOO trasponen los principios de la orientación a objetos a la tecnología de bases de datos y por ello son adecuadas sobre todo en la programación de aplicaciones orientada a objetos, pero los sistemas de bases de datos de este tipo son poco frecuentes y aún muy nuevos para el mercado.



**BBDD ENTIDAD - RELACIÓN**





# Bases de datos objeto-relacionales

Los sistemas mixtos son sistemas de bases de datos relacionales que se han ampliado con conceptos del modelo orientado a objetos. De este modo los principios probados del modelo relacional se ampliarían a tipos abstractos de datos como los objetos.

Para poder gestionar estos tipos abstractos de datos, las bases de datos objeto-relacionales amplían las bases de datos relationales con:

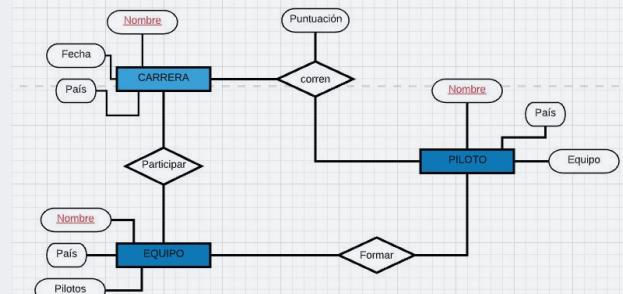
- Tipos de datos complejos y personalizables: mientras que las bases de datos relationales solo pueden procesar datos alfanuméricos, con los tipos de datos personalizables también se pueden gestionar archivos multimedia estructurados de forma compleja.
- Constructores de tipos: permiten衍生tipos nuevos de los tipos básicos ya existentes.
- Funciones y métodos: como SQL no permite crear funciones, los sistemas objeto-relacionales han de proveer extensiones con las cuales puedan definirse funciones de acceso y edición de tipos de datos complejos.

A partir del cambio de siglo se incluyen extensiones objeto-relacionales, como los tipos estructurados, en las últimas versiones del estándar SQL, pero no lo soportan todos los SGBD. Algunos sistemas conocidos que disponen de estas extensiones son **IBM Db2**, **Oracle Database** y **Microsoft SQL Server**.

```

1  {
2    "id" : 1,
3    "1er apellido" : "García",
4    "nombre" : "Antonio",
5    "nº SS" : " 32 12345678 12 ",
6    "calle" : "Calle principal, 1",
7    "CP" : "11111",
8    "Ciudad" : "Villarriba",
9    "coche_ID" : [1, 4]
10   }

```



## Bases de datos orientadas a documentos

Si en las bases de datos relationales los datos se almacenan en tablas, el modelo orientado a documentos se basa en un conjunto heterogéneo de datos compuesto por documentos, que pueden ser **estructurados**, como archivos **JSON**, **YAML** o **XML**, o **no estructurados**, como **Binary Large Objects (BLOB)** -archivos de imagen, vídeo o audio.

En el caso de los documentos estructurados, el almacenamiento tiene lugar por medio de pares clave/valor donde a cada clave se le asigna un valor concreto.

En este contexto se utiliza el término “clave” como sinónimo de atributo y no tiene nada que ver con las claves del modelo relacional. Los valores pueden equivaler a cualquier tipo de información. Las listas o las matrices también podrían ser valores.

Un documento en formato JSON para almacenar los datos de nuestra plantilla podría tener la construcción de la imagen.

Varios documentos pueden agruparse en una colección de documentos, una llamada **collection**. Este documento con datos de empleados podría unirse a otra parte de la colección “Empleados”.

Las consultas tienen lugar utilizando funciones, lo que puede hacerse con JavaScript. Los SGBD que trabajan orientados a documentos otorgan un ID único a cada documento, pero, a diferencia de como ocurre en el modelo relacional, **aquí no existe ningún esquema que abarque a la base de datos por completo**.



Los documentos en una base de datos basada en documentos no han de observar ninguna forma normal ni existen propiedades estructurales que hayan de cumplirse en todos los documentos. En principio, cada documento puede tener una estructura diferente. Pese a todo, en lo que hace al desarrollo de aplicaciones, se recomienda crear los documentos en uno de los esquemas adecuados a la aplicación para lograr los requisitos necesarios para poder realizar consultas.

Las relaciones, como la conexión de tablas de bases de datos en el modelo relacional, no son posibles aquí. Aunque es posible añadir manualmente el ID de un documento como referencia en un documento diferente, los SGBD orientados a documentos **no ofrecen JOIN**, con lo que estas posibilidades de consulta deben programarse a propósito.

En definitiva, los sistemas orientados a documentos son la mejor opción si se han de procesar **grandes cantidades de datos con estructura heterogénea y escasa interconexión**. Esto hace a este modelo apto sobre todo para el trabajo con big data.

Los sistemas relacionales controlan en todo momento que se cumplan las condiciones indicadas en las definiciones de las tablas, lo que, al procesar grandes volúmenes de datos, conduce a una menor velocidad de escritura. Los sistemas NoSQL no presentan unos requisitos de consistencia de datos tan estrictos, condición que los hace adecuados para grandes arquitecturas en las cuales se han de gestionar de forma paralela muchas instancias de base de datos.

También las aplicaciones web recurren con frecuencia creciente a las bases de datos orientadas a documentos. Pero si se requiere una fuerte interconexión, el almacenamiento basado en documentos va ligado a un esfuerzo mucho mayor. En este caso sería más conveniente utilizar sistemas relacionales.

Algunos ejemplos de bases de datos orientadas a documentos son **BaseX**, **CouchDB**, **eXist**, **MongoDB** y **RavenDB**.





# BBDD Relacional: ¿qué ventajas tienen?

El modelo relacional para bases de datos se ha impuesto, no sin motivo, en el entorno del procesamiento electrónico de datos. Resumimos a continuación los principales puntos fuertes de este modelo de base de datos:

## Sencillez:

El modelo de datos que subyace a la base de datos relacional se implementa y gestiona más fácilmente que otros modelos. Las ingentes cantidades de información (datos de clientes, listas de pedido, movimientos de las cuentas) que las empresas quieren almacenar a largo plazo se organizan sin problemas en la estructura de tablas en que se basa el modelo relacional de base de datos.

## Escasa redundancia de datos:

Las formas normales del modelo relacional fijan una normativa que tiene el fin de evitar duplicaciones. Si las reglas de normalización se aplican de forma consecuente, los sistemas relacionales facilitan un almacenamiento de datos libre de redundancias, puesto que solo es necesario editar los datos una única vez, lo que simplifica sobre todo el mantenimiento interno y técnico del banco de datos.

## Alta consistencia de datos:

Las bases de datos relacionales normalizadas permiten almacenar datos sin contradicciones, contribuyendo así a la consistencia de los datos. Asimismo, los sistemas relacionales presentan funciones con las cuales se definen y se controlan automáticamente las condiciones de integridad. Aquellas transacciones que ponen en peligro la consistencia de los datos se bloquean.

## Procesamiento de datos orientado a conjuntos:

El sistema de base de datos relacional se apoya en un procesamiento orientado a conjuntos que subdivide cada entidad en valores mínimos. Esto permite conectar entidades diferentes por medio del contenido, así como realizar consultas complejas como JOIN.



### Lenguaje de consultas homogéneo:

Para la realización de consultas a bases de datos relacionales se ha consolidado el lenguaje SQL, que ha sido estandarizado por la ISO y la IEC. El propósito de tal estandarización es que las aplicaciones puedan desarrollarse y ejecutarse con independencia del SGBD en que se utilicen. Con todo, el soporte de SQL varía mucho en función del SGBD.

## Inconvenientes de las bases de datos relacionales

Según el escenario en que se emplean los sistemas de bases de datos relacionales, ciertas ventajas, como el estar basados en tablas, así como el reparto de los datos en tablas interconectadas, pueden interpretarse también como desventajas. Además, algunas de sus propiedades más destacadas son difícilmente reconciliables con los modernos requisitos de la programación de aplicaciones (orientación a objetos, multimedia y big data).

### Presentación de los datos en tablas:

No siempre es posible integrar cualquier tipo de dato en el formato fijo de las tablas bidimensionales aun cuando estén interconectadas (*impedance mismatch*). Los datos abstractos o no estructurados que surgen en relación con las aplicaciones multimedia y las soluciones de big data no pueden representarse en el modelo relacional.

### Sistema no jerárquico:

Las bases de datos relacionales normalizadas permiten almacenar datos sin contradicciones, contribuyendo así a la consistencia de los datos. Asimismo, los sistemas relacionales presentan funciones con las cuales se definen y se controlan automáticamente las condiciones de integridad. Aquellas transacciones que ponen en peligro la consistencia de los datos se bloquean.

### Segmentación de los datos:

El principio de base de los sistemas relacionales que consiste en almacenar la información en tablas separadas (normalización) conduce inevitablemente a su segmentación. Este diseño deriva en complejas consultas que abarcan varias tablas, de modo que el elevado número de segmentos resultantes acostumbra a reflejarse negativamente en el rendimiento.



## Peor rendimiento frente a las bases de datos NoSQL:

*El modelo relacional plantea elevados requisitos en cuanto a la consistencia de datos que van en detrimento de la velocidad de escritura en las transacciones.*

## CONCLUSIÓN

*El modelo relacional para bases de datos se caracteriza por la claridad, tiene una base matemática y ha probado su eficacia en la práctica durante más de 40 años. Pese a todo, el almacenamiento de datos en tablas estructuradas no se ha adaptado a las necesidades de la tecnología de la información moderna.*

*Son especialmente la gestión de grandes volúmenes de datos en el marco de los análisis de big data y el almacenamiento de datos abstractos los factores que saturan la capacidad de los sistemas relacionales. Y es precisamente aquí donde los sistemas especializados, como las BD basadas en objetos o los conceptos desarrollados en la senda del movimiento NoSQL, muestran su superioridad, si bien no es posible prescindir completamente del modelo relacional.*

*Las bases de datos relacionales despliegan todo su potencial sobre todo en aquellos ámbitos corporativos protagonizados por el **procesamiento de datos de transacciones**.*

*Los datos sobre acciones de los clientes o medidas de marketing pueden representarse perfectamente en el formato de tabla, a la par que los usuarios sacan provecho de una sintaxis que, pese a su simplicidad, permite consultas complejas.*



# MySQL (Software)

---

**MySQL** es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general, junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

## ¿Qué es una base de datos MySQL?

---

**MySQL** es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Funciona con un modelo cliente-servidor. Eso quiere decir que los ordenadores que instalan y ejecutan el software de gestión de base de datos se denominan clientes.

Uno o más dispositivos (clientes) se conectan a un servidor a través de una red específica. Cada cliente puede realizar una solicitud desde la interfaz gráfica de usuario (GUI) en sus pantallas, y el servidor producirá el output deseado, siempre que ambas partes entiendan la instrucción. Sin meternos demasiado a fondo en temas técnicos, los procesos principales que tienen lugar en un entorno MySQL son los mismos, y son:

1. MySQL crea una base de datos para almacenar y manipular datos, definiendo la relación de cada tabla.
2. Los clientes pueden realizar solicitudes escribiendo instrucciones SQL específicas en MySQL.
3. La aplicación del servidor responderá con la información solicitada y esta aparecerá frente a los clientes.

Desde el lado de los clientes, generalmente enfatizan qué GUI de MySQL usar. Cuanto más ligera y fácil de usar sea la GUI, más rápidas y fáciles serán sus actividades de administración de datos. Algunas de las GUI de MySQL más populares son **MySQL WorkBench**, **SequelPro**, **DBVisualizer** y **Navicat DB Admin Tool**. Algunas de ellas son gratuitas, mientras que otras son comerciales, otras son exclusivamente para macOS y otras son compatibles con los principales sistemas operativos. Los clientes deben elegir la GUI en función de sus necesidades. Para la administración de bases de datos web, incluido un sitio de WordPress, la opción más obvia es **phpMyAdmin**.

## ¿Cómo se utiliza la base de datos MySQL?

---

**MySQL** crea una base de datos para almacenar y manipular datos, definiendo la relación de cada tabla. Los clientes pueden realizar solicitudes escribiendo instrucciones SQL específicas en MySQL. La aplicación del servidor responderá con la información solicitada y esta aparecerá frente a los clientes.



# Bases de datos MySQL: Ejercicios

Desde una importante cadena de tiendas de música, nos realizan el encargo de construir una base de datos que permita almacenar la siguiente información referente a los discos que venden:

- **Información de grupos y artistas.**
- **Información de productores, letristas, compositores y músicos participantes en la grabación.**
- **Información de los discos.**
- **Información de las canciones de los discos.**

Para poder enseñarle una primera versión al cliente con la intención de que nos contrate a nosotros en vez de a la competencia vamos a crear una base de datos pequeñita que mostrarle, el cliente nos ha dado dos semanas de tiempo.

**Realiza las siguientes tareas a la hora de crear la base de datos.**

**Realiza un diseño de base de datos que cubra las necesidades planteadas por el cliente.**

- Crea la base de datos y dale por nombre "music".
- Crea las tablas de la base de datos en base al diseño que has planteado previamente.
- Inserta en las tablas correspondientes la información incluida en el anexo referente a los siguientes discos:
  - "Tentación" de La Unión.
  - "Still got the blues" de Gary Moore.
  - "After hors" de Gary Moore.
- Añade también algún disco que te guste.

**Realiza las consultas de selección para recuperar la siguiente información:**

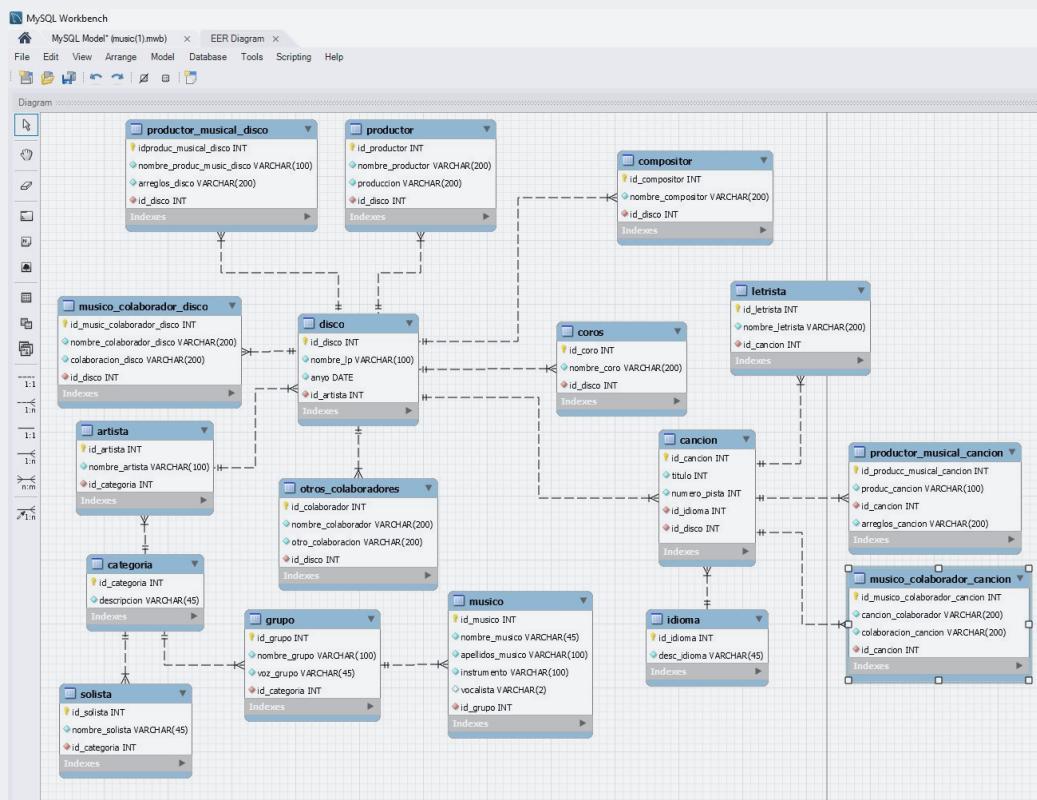
- Recupera la información de todos los artistas registrados en la base de datos por orden alfabético.
- Recupera por orden alfabético la información de todos los discos registrados.
- Recupera por orden cronológico todos los discos publicados por el músico Gary Moore que están registrados en la base de datos.
- Recupera todas las canciones del elepé "Tentación" tal y como aparecen ordenadas en el disco.
- Recupera toda la información relevante de la canción "Walking by myself".
- Modifica la estructura de alguna de las tablas con el objeto de poder almacenar también las letras de las canciones.
- Incluye la letra de la canción "Walking by myself" en la base de datos. La encontrarás en el anexo.
- Elimina la canción bonustrack del elepé "Still got the blues".
- Realiza una copia de seguridad de la base de datos.
- Elimina la base de datos.
- Recupera la copia de seguridad de la base de datos.
- Crea una vista que te permita recuperar toda la información relevante de la canción "Walking by myself".



Para la realización de un primer diseño de la base de datos, me he basado en los siguientes criterios:

- **Composer:** Persona que escribe todas las canciones.
- **Letrista:** Persona que escribe una canción.
- **Productor:**
  - **Productor ejecutivo**
  - **Productor musical:** Realiza arreglos instrumentales y vocales, tanto en el disco como en una canción.
  - **Artista:** Artista principal del Disco, solista o grupo.

- Realiza un diseño de base de datos que cubra las necesidades planteadas por el cliente.



- Crea la base de datos y dale por nombre “music”.

```
CREATE DATABASE IF NOT EXISTS music;
```

o

```
CREATE SCHEMA IF NOT EXISTS music;
```



- Crea las tablas de la base de datos en base al diseño que has planteado previamente.

Tabla	Acción
artista	Examinar Estructura Buscar Insertar Vaciar Eliminar
cancion	Examinar Estructura Buscar Insertar Vaciar Eliminar
categoria	Examinar Estructura Buscar Insertar Vaciar Eliminar
compositor	Examinar Estructura Buscar Insertar Vaciar Eliminar
coros	Examinar Estructura Buscar Insertar Vaciar Eliminar
disco	Examinar Estructura Buscar Insertar Vaciar Eliminar
grupo	Examinar Estructura Buscar Insertar Vaciar Eliminar
idioma	Examinar Estructura Buscar Insertar Vaciar Eliminar
letrista	Examinar Estructura Buscar Insertar Vaciar Eliminar
musico	Examinar Estructura Buscar Insertar Vaciar Eliminar
musico_colaborador_cancion	Examinar Estructura Buscar Insertar Vaciar Eliminar
musico_colaborador_disco	Examinar Estructura Buscar Insertar Vaciar Eliminar
otros_colaboradores	Examinar Estructura Buscar Insertar Vaciar Eliminar
productor	Examinar Estructura Buscar Insertar Vaciar Eliminar
productor_musical_cancion	Examinar Estructura Buscar Insertar Vaciar Eliminar
productor_musical_disco	Examinar Estructura Buscar Insertar Vaciar Eliminar
solista	Examinar Estructura Buscar Insertar Vaciar Eliminar

Número de filas: 17

- Inserta en las tablas correspondientes la información incluida en el anexo referente a los siguientes discos:

**Sentencia tipo INSERT:**

```
INSERT INTO nombre_tabla
    FROM (nombre_campo1, nombre_campo2,...)
    VALUES (valor_campo1, valor_campo2, vacío (null));
```

- “Tentación” de La Unión.
- “Still got the blues” de Gary Moore.
- “After hours” de Gary Moore.
- Añade también algún disco que te guste.

			id_disco	nombre_lp	año	id_artista
<input type="checkbox"/>	Editar  Copiar  Borrar	1	Tentación	1990	1	
<input type="checkbox"/>	Editar  Copiar  Borrar	2	Still got the blues	1990	2	
<input type="checkbox"/>	Editar  Copiar  Borrar	3	After hours	1992	2	
<input type="checkbox"/>	Editar  Copiar  Borrar	4	Miguel Bosé	1978	3	



- Realiza las consultas de selección para recuperar la siguiente información.
  - Recupera la información de todos los artistas registrados en la base de datos por orden alfabético.

**Sentencia tipo SELECT:**

```
SELECT
    a.nombre_artista AS Artista,
    c.descripcion AS Descripción
FROM
    artista a,
    categoria c
WHERE
    a.id_categoria = c.id_categoria
ORDER BY
    a.nombre_artista ASC;
```

SELECT a.nombre_artista AS Artista, c.descripcion AS Descripción FROM artista a, categoria c WHERE a.id_categoria = c.id_categoria ORDER BY a.nombre_artista ASC	
<input type="checkbox"/> Mostrar todo	Número de filas: 25
Filtrar filas: <input type="text"/> Buscar en esta tabla	
+ Opciones	
Artista	Descripción
Gary Moore	solista
La Unión	grupo
Miguel Bosé	solista

- Recupera por orden alfabético la información de todos los discos registrados.

```
SELECT
    d.nombre_lp AS Álbum,
    d.anho AS Publicación,
    a.nombre_artista AS Artista,
    compositor.nombre_compositor AS compositor,
    p.nombre_productor AS Productor,
    c.nombre_coro AS COROS,
    CONCAT(CAST(m.nombre_colaborador_disco AS VARCHAR (15)), ',',
           CAST(m.colaboracion_disco AS VARCHAR(20))) AS Colaboración,
    CONCAT(CAST(producc.nombre_produc_music_disco AS VARCHAR (10)), ',',
           CAST(producc.arreglos_disco AS VARCHAR (20))) AS Arreglos,
    CONCAT(CAST(colaborador.nombre_colaborador AS VARCHAR(15)), ',',
           CAST(colaborador.otro_colaboracion AS VARCHAR (20))) AS Otros
FROM disco d
INNER JOIN artista a ON (d.id_artista = a.id_artista)
LEFT JOIN compositor compositor ON (d.id_disco = compositor.id_disco)
LEFT JOIN productor p ON (d.id_disco = p.id_disco)
LEFT JOIN coros c ON (d.id_disco = c.id_disco)
LEFT JOIN musico_colaborador_disco m ON (d.id_disco = m.id_disco)
LEFT JOIN productor_musical_disco producc ON (d.id_disco = producc.id_disco)
LEFT JOIN otros_colaboradores colaborador ON (d.id_disco = colaborador.id_disco);
```



## → Con GROUP BY se reduce mucho la información.

```
SELECT
    d.nombre_lp AS Álbum,
    d.anio AS Publicación,
    a.nombre_artista AS Artista,
    compositor.nombre_compositor AS compositor,
    p.nombre_productor AS Productor,
    coro.nombre_coro AS COROS,
    CONCAT(CAST(m.nombre_colaborador_disco AS VARCHAR(15)), ',',
           CAST(m.colaboracion_disco AS VARCHAR(20))) AS Colaboración,
    CONCAT(CAST(produc.nombre_produc_music_disco AS VARCHAR(10)), ',',
           CAST(produc.arreglos_disco AS VARCHAR(20))) AS Arreglos,
    CONCAT(CAST(colaborador.nombre_colaborador AS VARCHAR(15)), ',',
           CAST(colaborador.otro_colaboracion AS VARCHAR(20))) AS Otros
FROM disco d
INNER JOIN artista a ON (d.id_artista = a.id_artista)
LEFT JOIN compositor compositor ON (d.id_disco = compositor.id_disco)
LEFT JOIN productor p ON (d.id_disco = p.id_disco)
LEFT JOIN coros coro ON (d.id_disco = coro.id_disco)
LEFT JOIN musico_colaborador_disco m ON (d.id_disco = m.id_disco)
LEFT JOIN productor_musical_disco produc ON (d.id_disco = produc.id_disco)
LEFT JOIN otros_colaboradores colaborador ON (d.id_disco = colaborador.id_disco)

GROUP BY
    d.nombre_lp ASC;
```

- Recupera por orden cronológico todos los discos publicados por el músico Gary Moore que están registrados en la base de datos.

A) Teniendo en cuenta que sabemos el id del artista, la sentencia sería:

```
SELECT nombre_lp AS Lp, anio AS Publicación
FROM disco
WHERE id_artista = 2
ORDER BY anio ASC;
```

SELECT nombre_lp AS Lp, anio AS Publicación FROM disco WHERE id_artista = 2 ORDER BY anio ASC		
<input type="checkbox"/> Mostrar todo	Número de filas: 25	Filtrar filas: Buscar en esta tabla
+ Opciones	Lp	Publicación
<input type="checkbox"/> Editar <input checked="" type="checkbox"/> Copiar <input checked="" type="checkbox"/> Borrar Still got the blues	1990	
<input type="checkbox"/> Editar <input checked="" type="checkbox"/> Copiar <input checked="" type="checkbox"/> Borrar After hours	1992	



B) con SUBCONSULTAS, la sentencia sería:

```
SELECT
    nombre_lp AS Lp,
    anyo AS Publicación
FROM disco
WHERE id_artista = (
    SELECT id_artista FROM artista WHERE nombre_artista LIKE 'Gary MOORE'
)
ORDER BY anyo ASC;
```

SELECT nombre_lp AS Lp, anyo AS Publicación FROM disco WHERE id_artista = ( SELECT id_artista FROM artista WHERE nombre_artista LIKE 'Gary MOORE' ) ORDER BY anyo ASC			
<input type="checkbox"/> Mostrar todo	Número de filas: 25	Filtrar filas: Buscar en esta tabla	Ordenar según la clave: Ninguna
<b>Opciones</b> <input type="button" value="←"/> <input type="button" value="→"/> Lp Publicación			
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Still got the blues	1990	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> After hours	1992	

- Recupera todas las canciones del elepé “Tentación” tal y cómo aparecen ordenadas en el disco.

A) una forma sería:

```
SELECT titulo AS 'Nombre Canción', numero_pista AS Pista
FROM cancion
WHERE
    ( id_disco = id_disco )
    AND
    id_disco = 1;
```

SELECT titulo AS 'Nombre Canción', numero_pista AS Pista FROM cancion WHERE id_disco = id_disco AND id_disco=1			
<input type="checkbox"/> Mostrar todo	Número de filas: 25	Filtrar filas: Buscar en esta tabla	Ordenar según la clave:
<b>Opciones</b> <input type="button" value="←"/> <input type="button" value="→"/> Nombre Canción Pista			
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Tentación	1	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Fueron los celos	2	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Ella es un volcán	3	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Si tú quisieras	4	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Dámelo ya	5	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Llámame	6	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Revolución	7	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Más dura será la caída	8	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Nana	9	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Santa María	10	
<input type="checkbox"/>	<input type="button" value="Edit"/> Copiar <input type="button" value="Borrar"/> Berlín	11	



B) Otra forma de hacerlo sería:

```
SELECT
    d.nombre_lp AS Álbum,
    c.título AS Canción,
    c.numero_pista AS Pista
FROM disco d
JOIN canción c
ON (d.id_disco = c.id_disco)
WHERE d.nombre_lp LIKE '%Tenta%';
```

SELECT d.nombre_lp AS Álbum, c.título AS Canción, c.numero_pista AS Pista FROM disco d JOIN canción c ON (d.id_disco = c.id_disco) WHERE d.nombre_lp LIKE '%Tenta%'		
<input type="checkbox"/> Mostrar todo	Número de filas: 25	Filtrar filas: Buscar en esta tabla
Ordenar según la clave: Ninguna		
+ Opciones		
Álbum	Canción	Pista
Tentación	Tentación	1
Tentación	Fueron los celos	2
Tentación	Ella es un volcán	3
Tentación	Si tú quisieras	4
Tentación	Dámelo ya	5
Tentación	Llámame	6
Tentación	Revolución	7
Tentación	Más dura será la caída	8
Tentación	Nana	9
Tentación	Santa María	10
Tentación	Berlín	11

- Recupera toda la información relevante de la canción “Walking by myself”.

```
(SELECT
    c.título AS Canción,
    c.numero_pista AS Pista,
    i.desc_idioma AS Idioma,
    d.nombre_lp AS Álbum,

    CONCAT(CAST(l.nombre_letrista AS CHAR(6)), ' ',
           CAST(l.apellidos_letrista AS CHAR(6))) AS Letrista,
           m.cancion_colaborador AS Colaborador,
           m.colaboracion_cancion AS Colaboración

FROM canción c
INNER JOIN idioma i ON (i.id_idioma = c.id_idioma)
INNER JOIN disco d ON (d.id_disco = c.id_disco)
INNER JOIN letrista l ON (l.id_cancion = c.id_cancion)
LEFT JOIN musico_colaborador_cancion m ON (m.id_cancion = c.id_cancion)

WHERE
    (c.título = 'Walking by myself')
);
```

(SELECT c.título AS Canción, c.numero_pista AS Pista, i.desc_idioma AS Idioma, d.nombre_lp AS Álbum, CONCAT(CHAR(6), ' ', CHAR(6)) AS Letrista, m.cancion_colaborador AS Colaborador, m.colaboracion_cancion AS Colaboración FROM canción c INNER JOIN idioma i ON (i.id_idioma = c.id_idioma) INNER JOIN disco d ON (d.id_disco = c.id_disco) INNER JOIN letrista l ON (l.id_cancion = c.id_cancion) LEFT JOIN musico_colaborador_cancion m ON (m.id_cancion = c.id_cancion) WHERE (c.título = 'Walking by myself'))						
<a href="#">[Editar en línea]</a> <a href="#">[Editar]</a> <a href="#">[Crear código PHP]</a>						
<input type="checkbox"/> Mostrar todo	Número de filas: 25	Filtrar filas: Buscar en esta tabla	Ordenar según la clave: Ninguna			
Canción	Pista	Idioma	Álbum	Letrista	Colaborador	Colaboración
Walking by myself	3	Inglés	Still got the blues	Jimmy Rogers	Gary Moore	Guitarra/voz
Walking by myself	3	Inglés	Still got the blues	Jimmy Rogers	Mick Weaver	Piano
Walking by myself	3	Inglés	Still got the blues	Jimmy Rogers	Andy Pyle	Bajo
Walking by myself	3	Inglés	Still got the blues	Jimmy Rogers	Graham Walker	Tambor
Walking by myself	3	Inglés	Still got the blues	Jimmy Rogers	Frank Mead	Armónica



- Modifica la estructura de alguna de las tablas con el objeto de poder almacenar también las letras de las canciones.

```
USE music;
ALTER TABLE cancion ADD COLUMN letra text AFTER titulo;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_cancion	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar  Más
2	titulo	varchar(200)	utf8mb4_general_ci		No	Ninguna			Cambiar  Eliminar  Más
3	letra	text	utf8mb4_general_ci		Sí				Cambiar  Eliminar  Más
4	numero_pista	int(11)			No	Ninguna			Cambiar  Eliminar  Más
5	id_idioma	int(11)			No	Ninguna			Cambiar  Eliminar  Más
6	id_disco	int(11)			Sí	NULL			Cambiar  Eliminar  Más

- Incluye la letra de la canción “Walking by myself” en la base de datos. La encontrarás en el anexo.

#### 1) Obtener el id\_cancion de la canción a actualizar.

```
USE music;
SELECT c.id_cancion, c.titulo
FROM cancion
WHERE (c.titulo = 'Walking by myself');
```

```
SELECT c.id_cancion, c.titulo FROM cancion c WHERE (c.titulo = 'Walking by myself')
```

Mostrar todo	Número de filas:	25	Filtrar filas: Buscar en esta tabla
+ Opciones		id_cancion	titulo
<input type="checkbox"/> Editar  Copiar  Borrar	14	Walking by myself	

#### 2) Realizar la actualización del campo letra.

```
USE music;
UPDATE cancion c
```

```
SET c.letra = 'You know I love you. You know it's true. Give you all my love, babe. What more can I do? Walking by myself, I hope you'll understand. I just want to be your lovin' man. I love ya, yes I love you with my heart and soul. I wouldn't mistreat you for my weight in gold. You know I love you. You know it's true. Give you all my love, babe. What more can I do? Walking by myself, I hope you'll understand. I just want to be your lovin' man. Here we go! Keep on walkin!! You know I love you. You know it's true. I give you all my, babe. What more can I do? I'm walking by myself, I hope you'll understand. I just want to be your lovin' man. I said I'm walking by myself, I hope you'll understand. I just want to be your lovin', I just want to be your lovin', I just want to be your lovin' man. That's right.'
```

```
WHERE c.id_cancion = 14;
```



SELECT * FROM `cancion` WHERE id_cancion = 14							
<input type="checkbox"/> Mostrar todo   Número de filas: 25 ▾ Filtrar filas: Buscar en esta tabla							
+ Opciones	← →	id_cancion	titulo	letra	numero_pista	id_idioma	id_disco
<input type="checkbox"/> Editar  Copiar  Borrar		14	Walking by myself	You know I love you. You know it's true. Give you ...	3	2	2

- Elimina la canción bonustrack del elepé “Still got the blues”.

```
USE music;
```

- Borrar los índices de las claves foráneas.

```
ALTER TABLE cancion DROP INDEX cancion_disco_fk;
ALTER TABLE letrista DROP INDEX letrista_cancion_fk;
ALTER TABLE productor_musical_cancion DROP INDEX producc_music_cancion_fk;
ALTER TABLE musico_colaborador_cancion DROP INDEX music_colabora_cancion_fk;
```

- Borrar todas las Claves foráneas relacionadas con la tabla cancion, de lo contrario no permite borrar datos.

```
ALTER TABLE cancion DROP FOREIGN KEY cancion_disco_fk;
ALTER TABLE letrista DROP FOREIGN KEY letrista_cancion_fk;
ALTER TABLE productor_musical_cancion DROP FOREIGN KEY producc_music_cancion_fk;
ALTER TABLE musico_colaborador_cancion DROP FOREIGN KEY music_colabora_cancion_fk;
```

- Borrar la canción bonustrack

```
DELETE FROM cancion WHERE titulo='bonustrack';
```

### Comprobación.

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas).

```
SELECT * FROM cancion WHERE titulo='bonustrack'
```

id_cancion	titulo	letra	numero_pista	id_idioma	id_disco
------------	--------	-------	--------------	-----------	----------



- Realiza una copia de seguridad de la base de datos.

Entramos en la base de datos music.  
Haciendo clic en el enlace, parte superior del menú,

Base de datos:

En el menú superior de la ventana en la que nos encontramos.  
Hacer clic en el botón exportar.

Exportando tablas de la base de datos "music"

Método de exportación:

Rápido - mostrar sólo el mínimo de opciones de configuración  
 Personalizado - mostrar todas las opciones de configuración posibles

Formato:

SQL

Continuar

Desde la pestaña FORMATO, podemos seleccionar el formato del archivo a descargar.



#### Método de exportación:

- Rápido - mostrar sólo el mínimo de opciones de configuración  
 Personalizado - mostrar todas las opciones de configuración posibles

#### Formato:

SQL

#### Tablas:

	Tablas	Estructura	Datos
<input type="checkbox"/>	Seleccionar todo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	artista	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	cancion	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	cancion_vista_view	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	categoria	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	compositor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	coros	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	disco	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	grupo	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Desde esta pestaña podemos personalizar el método de exportación.

#### Opciones de creación de datos

Truncar tablas antes de insertar

Sentencias INSERT DELAYED

Sentencias INSERT IGNORE

Función a utilizar al volcar datos:

Sintáxis a utilizar al insertar datos:

incluir nombres de columna en toda sentencia INSERT  
Ejemplo: `INSERT INTO nombre_tabla (columna_A,columna_B,columna_C) VALUES (1,2,3)`

incluir múltiples filas en cada sentencia INSERT  
Ejemplo: `INSERT INTO nombre_tabla VALUES (1,2,3), (4,5,6), (7,8,9)`

los dos anteriores  
Ejemplo: `INSERT INTO nombre_tabla (columna_A,columna_B,columna_C) VALUES (1,2,3), (4,5,6), (7,8,9)`

ninguno de los anteriores  
Ejemplo: `INSERT INTO nombre_tabla VALUES (1,2,3)`

Longitud máxima de la consulta creada

Volcar columnas binarias en notación hexadecimal (*por ejemplo "abc" sería 0x616263*)

Volcar columnas TIMESTAMP en UTC (*habilita que las columnas TIMESTAMP sean volcadas y cargadas por servidores en diferentes zonas horarias*)

Desde esta pestaña podemos seleccionar las sentencias que queremos que aparezcan en el archivo a descargar.

Podemos personalizar el volcado de datos.

Una vez personalizado o no, para que empiece la descarga, hacemos clic en continuar, para que se genere la copia de seguridad de la base de datos music.



- Recupera la copia de seguridad de la base de datos.

Accedemos al panel de control de phpMyAdmin



Hacer clic en el botón importar.

#### Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.  
Un archivo comprimido tiene que terminar en [formato].[compresión]. Por ejemplo: .sql.zip

Buscar en su ordenador  Ningún archivo seleccionado (Máximo: 40MB)

También puede arrastrar un archivo en cualquier página.

Conjunto de caracteres del archivo:

Seleccionamos el archivo que queremos importar.

#### Formato:

Desde esta pestaña podemos seleccionar el tipo de formato que queremos importar.

#### Formato:

#### Opciones específicas al formato:

Modalidad SQL compatible:

No utilizar AUTO\_INCREMENT con el valor 0

Hacemos clic en continuar.



Servidor: 127.0.0.1 > Base de datos: music

Estructura SQL Buscar Generar una consulta Exportar Importar Operaciones Privilegios Rutinas Eventos Disparadores

Filtros

Que contengan la palabra:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
artista	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-
cancion	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	43	InnoDB	utf8mb4_general_ci	48.0 KB	-
cancion_vista_view	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	~8	Visualizar	---	-	-
categoria	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	16.0 KB	-
compositor	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_general_ci	32.0 KB	-
coros	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	7	InnoDB	utf8mb4_general_ci	32.0 KB	-
disco	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
grupo	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_general_ci	32.0 KB	-
idioma	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	16.0 KB	-
letrista	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	40	InnoDB	utf8mb4_general_ci	32.0 KB	-
musico	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	32.0 KB	-
musico_colaborador_cancion	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	79	InnoDB	utf8mb4_general_ci	32.0 KB	-
musico_colaborador_disco	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	36	InnoDB	utf8mb4_general_ci	32.0 KB	-
otros_colaboradores	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	20	InnoDB	utf8mb4_general_ci	32.0 KB	-
productor	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	32.0 KB	-
productor_musical_cancion	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	26	InnoDB	utf8mb4_general_ci	32.0 KB	-
productor_musical_disco	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	14	InnoDB	utf8mb4_general_ci	32.0 KB	-
solista	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_general_ci	32.0 KB	-
18 tablas	Número de filas	~288	InnoDB	utf8mb4_general_ci	528.0 KB	0 B

Hemos recuperado la copia de seguridad de la base de datos.

- **Crea una vista que te permita recuperar toda la información relevante de la canción “Walking by myself”.**

```
CREATE VIEW cancion_vista_view AS
SELECT
    c.titulo AS Canción,
    c.numero_pista AS Pista,
    i.desc_idioma AS Idioma,
    d.nombre_lp AS Álbum,
    CONCAT( CAST(l.nombre_letrista AS CHAR(6)),
    ' ',
    CAST(l.apellidos_letrista AS CHAR(6))) AS Letrista,
    m.cancion_colaborador AS Colaborador,
    m.colaboracion_cancion AS Colaboración
FROM cancion c
    INNER JOIN idioma i ON (i.id_idioma = c.id_idioma)
    INNER JOIN disco d ON (d.id_disco = c.id_disco)
    INNER JOIN letrista l ON (l.id_cancion = c.id_cancion)
    LEFT JOIN ( musico_colaborador_cancion m ON (m.id_cancion = c.id_cancion)
WHERE
    (c.titulo = 'Walking by myself');
```



music

Tablas

- Nueva
- artista
- cancion
- categoria
- compositor
- coros
- disco
- grupo
- idioma
- letrista
- musico
- musico\_colaborador\_cancion
- musico\_colaborador\_disco
- otros\_colaboradores
- productor
- productor\_musical\_cancion
- productor\_musical\_disco
- solistas

Vistas

- Nueva
- cancion\_vista\_view

Perfiles

SELECT \* FROM `cancion\_vista\_view`

	Canción	Pista	Idioma	Álbum	Letrista	Colaborador	Colaboración	letra
<input type="checkbox"/>	Walking by myself	3	inglés	Still got the blues	Jimmy Rogers	Gary Moore	Guitarra/voz	You know I love you. You know it's true. Give you ...
<input type="checkbox"/>	Walking by myself	3	inglés	Still got the blues	Jimmy Rogers	Mick Weaver	Piano	You know I love you. You know it's true. Give you ...
<input type="checkbox"/>	Walking by myself	3	inglés	Still got the blues	Jimmy Rogers	Andy Pyle	Bajo	You know I love you. You know it's true. Give you ...
<input type="checkbox"/>	Walking by myself	3	inglés	Still got the blues	Jimmy Rogers	Graham Walker	Tambores	You know I love you. You know it's true. Give you ...
<input type="checkbox"/>	Walking by myself	3	inglés	Still got the blues	Jimmy Rogers	Frank Mead	Armónica	You know I love you. You know it's true. Give you ...

Opciones

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla

Editar Copiar Borrar Walking by myself 3 inglés Still got the blues Jimmy Rogers Gary Moore Guitarra/voz You know I love you. You know it's true. Give you ...

Editar Copiar Borrar Walking by myself 3 inglés Still got the blues Jimmy Rogers Mick Weaver Piano You know I love you. You know it's true. Give you ...

Editar Copiar Borrar Walking by myself 3 inglés Still got the blues Jimmy Rogers Andy Pyle Bajo You know I love you. You know it's true. Give you ...

Editar Copiar Borrar Walking by myself 3 inglés Still got the blues Jimmy Rogers Graham Walker Tambores You know I love you. You know it's true. Give you ...

Editar Copiar Borrar Walking by myself 3 inglés Still got the blues Jimmy Rogers Frank Mead Armónica You know I love you. You know it's true. Give you ...

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

Comprobamos que se haya creado.

← Servidor: 127.0.0.1 » Base de datos: music » Visualizar: cancion\_vista\_view

Examinar Estructura SQL Buscar Insertar Exporta

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
<input type="checkbox"/>	1 Canción	varchar(200)	utf8mb4_general_ci		No	Ninguna
<input type="checkbox"/>	2 Pista	int(11)			No	Ninguna
<input type="checkbox"/>	3 Idioma	varchar(45)	utf8mb4_general_ci		No	Ninguna
<input type="checkbox"/>	4 Álbum	varchar(100)	utf8mb4_general_ci		No	Ninguna
<input type="checkbox"/>	5 Letrista	varchar(14)	utf8mb4_general_ci		Sí	NULL
<input type="checkbox"/>	6 Colaborador	varchar(100)	utf8mb4_general_ci		Sí	NULL
<input type="checkbox"/>	7 Colaboración	varchar(200)	utf8mb4_general_ci		Sí	NULL
<input type="checkbox"/>	8 letra	text	utf8mb4_general_ci		Sí	

↑ Seleccionar todo Para los elementos que están marcados: Examinar

Editar vista Imprimir

Desde la pestaña vistas, podemos visualizar su estructura.