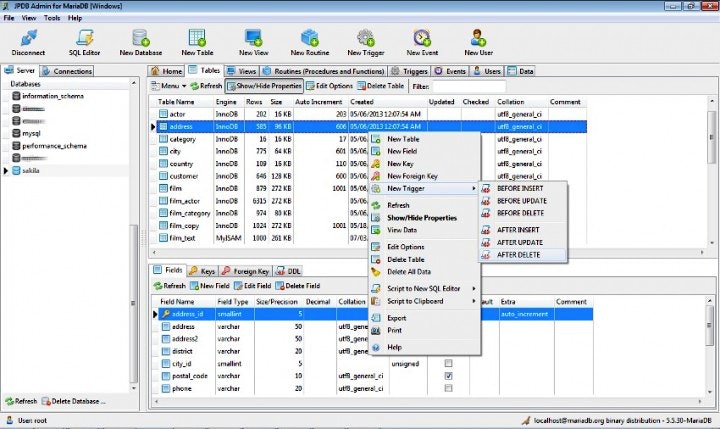
**SQL**

[MariaDB](https://www.capterra.es/software/132005/mariadb)



Esta solución, creada por los desarrolladores originales de MySQL, es empleada por gigantes tecnológicos como Wikipedia, Facebook e incluso Google. MariaDB es un servidor de base de datos que ofrece funcionalidad de sustitución directa para MySQL. La seguridad es prioritaria para los desarrolladores de MariaDB, que incluyen en cada versión los parches de seguridad de MySQL e incluso los mejoran si es necesario.

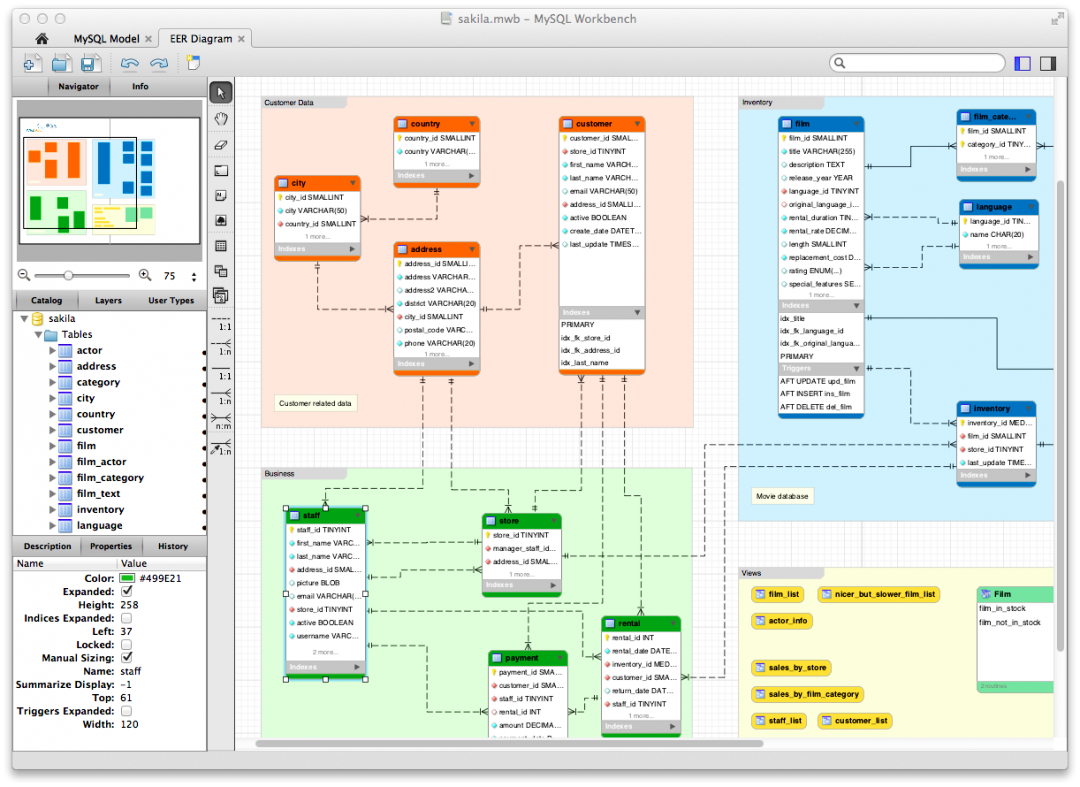
Ventajas

* Alta escalabilidad y fácil integración
* Acceso en tiempo Real
* Funcionalidad central de MySQL
  + (MariaDB es una alternativa a MySQL)
* Motores de almacenamiento alternativos, optimizaciones de servidor y parches
* Completa base de conocimiento, compilada a lo largo de los 20 años de funcionamiento de MariaDB

Desventajas

* Carece de plugin de complejidad de contraseña
* Sin interfaz en caché de memoria
* Sin rastreador de optimización

[MySQL](https://www.capterra.es/software/11482/mysql)



MySQL lleva en el mercado desde 1995 y ahora es propiedad de Oracle. Aunque es de código abierto, dispone de varias ediciones de pago que ofrecen funciones adicionales, como georreplicación de clústeres y escalado automático. Ya que MySQL, escrito tanto en C como en C++, es un estándar del sector, es compatible con prácticamente cualquier sistema operativo. Esta solución es ideal para usuarios internacionales, ya que el servidor puede ofrecer a los clientes mensajes de error en múltiples idiomas y cuenta con soporte para distintos conjuntos de caracteres.

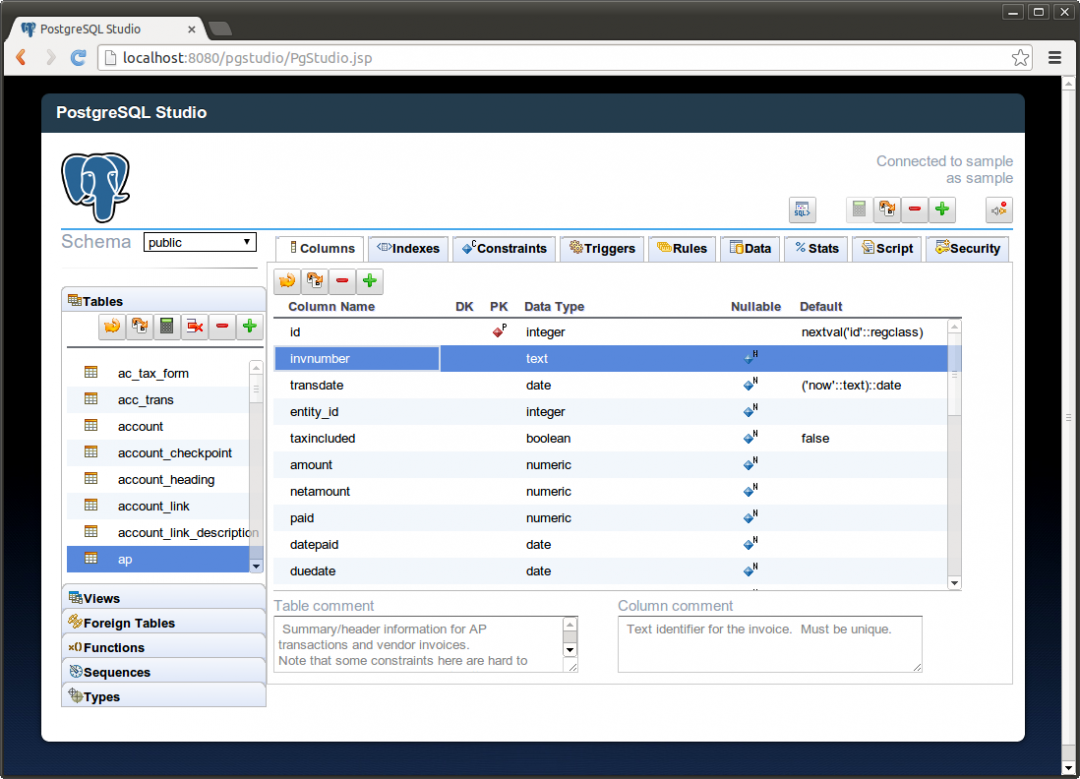
Ventajas

* Verificación basada en host
* Puede usarse aunque no haya ninguna red disponible
* Sistema flexible de privilegios y contraseñas
* Cifrado de seguridad de todo el tráfico de contraseñas
* La biblioteca se puede integrar en aplicaciones independientes
* Ofrece el servidor por separado para entornos con redes de cliente/servidor

Desventajas

* Adquirida por Oracle:
  + Los usuarios sienten que MySQL ya no es gratuita y de código abierto.
  + Ya no es una solución impulsada por la comunidad.
  + Los usuarios no pueden solucionar errores ni crear parches.
  + Superada por otras soluciones que se actualizan con mayor frecuencia.

[PostgreSQL](https://www.capterra.es/software/48791/postgresql)



Con más de 15 años de desarrollo activo, PostgreSQL es otra de las opciones más importantes de código abierto. Puede ejecutarse en todos los principales sistemas operativos, incluidos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. PostgreSQL también cumple las normas ACID (atomicidad, consistencia, aislamiento, durabilidad).

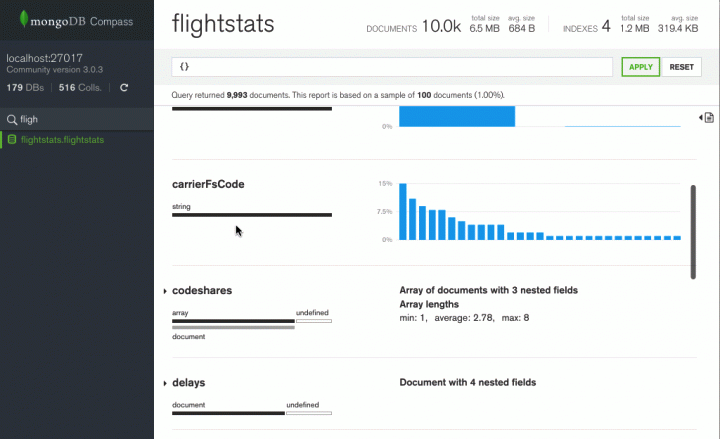
Ventajas

* Creación de tipos de datos y métodos de consulta personalizados
* El framework permite definir y crear tipos de datos personalizados
* Ejecuta procedimientos almacenados en más de una docena de lenguajes de programación:
  + Java, Perl, Python, Ruby, Tcl, C/C++ y su propio sistema
* GiST (árbol de búsqueda generalizado) PL/pgSQL
  + Reúne distintos algoritmos de ordenación y búsqueda:
    - árbol B-, árbol B+-, árbol R-, árboles de suma parcial y árboles de rango B+-
* Creación de extensiones, como CitusDB, para un mayor paralelismo sin necesidad de modificar el código de Postgres

Desventajas

* El sistema MVCC requiere de una “limpieza” frecuente
  + Problemas en entornos con altas tasas de transacción
* Es la comunidad quien realiza el desarrollo
  + Las mejoras añadidas conllevan mucho trabajo

[MongoDB](https://www.capterra.es/software/127374/mongodb)



MongoDB, fundada en 2007, se conoce como “la base de datos de las ideas gigantes”. Desarrollada por el equipo que creó DoubleClick, ShopWiki y Gilt Groupe, MongoDB cuenta con el respaldo de conocidos inversores como Fidelity Investments, The Goldman Sachs Group, Inc. e Intel Capital. Desde su creación, MongoDB se ha descargado 20 millones de veces y cuenta con más de 1 000 socios dedicados a esta solución gratuita y de código abierto que abogan por bases de datos y programaciones sencillas y naturales.

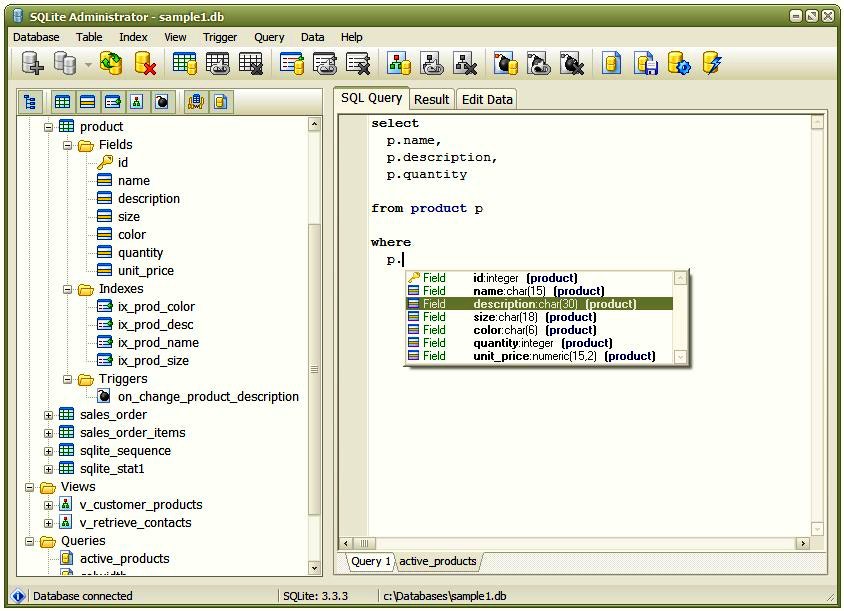
Ventajas

* Validación de documentos
* Motor de almacenamiento cifrado
* Casos de uso comunes:
  + Aplicaciones móviles
  + Catálogos de producto
  + [Gestión de contenido](https://www.capterra.es/directory/10007/content-management/software)
* Aplicaciones en tiempo real con motor de almacenamiento en memoria (beta)
* Tiempo reducido entre fallo primario y recuperación

Desventajas

* No es adecuada para aplicaciones que necesiten de transacciones complejas
* No es adecuada para sustitución directa de aplicaciones heredadas
* Solución joven: el software cambia y evoluciona rápidamente

[SQLite](https://www.capterra.es/software/158187/sqlite)



SQLite, autoproclamada como “la base de datos más implementada del mundo”, se creó en 2000 y ha contado entre sus usuarios con importantes empresas como Apple, Facebook, Microsoft y Google. Cada versión se prueba minuciosamente para garantizar su fiabilidad. En caso de errores, los desarrolladores de SQLite indican honestamente las potenciales desventajas, ofreciendo en cada versión listas de errores y cronologías de cambios en el código.

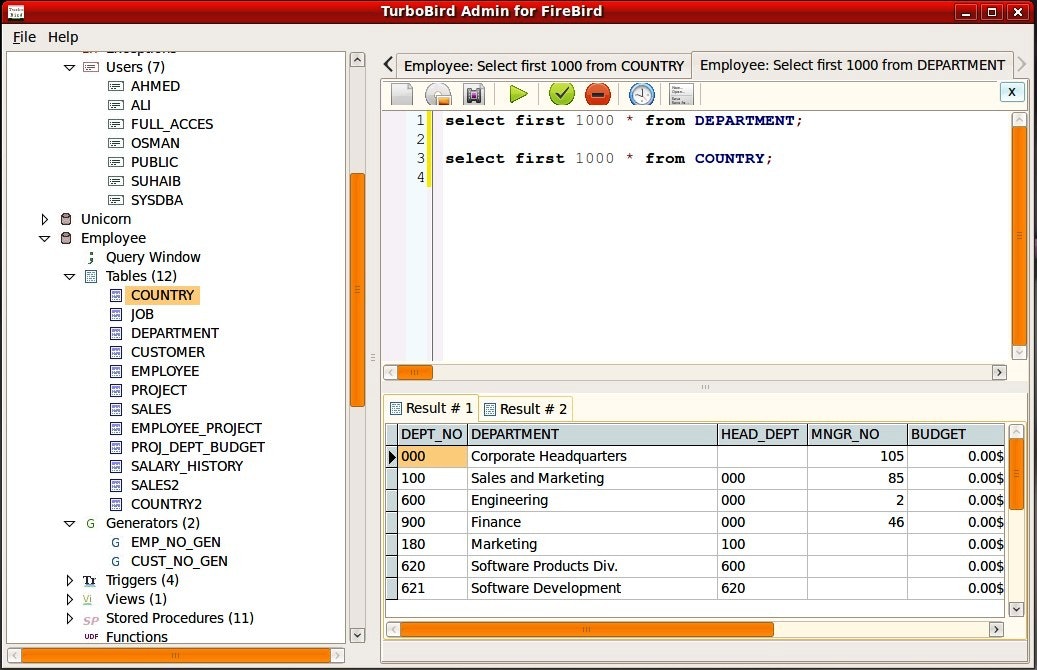
Ventajas

* Sin procesos de servidor por separado
* Formato de archivos multiplataforma
* Biblioteca compacta: se ejecuta más rápido incluso con más memoria
* Transacciones conformes a ACID
* Ofrece soporte profesional

Desventajas

* No se recomienda para:
  + Aplicaciones cliente/servidor
  + Webs con alto volumen
  + Grandes conjuntos de datos
  + Alta concurrencia

[Firebird](https://www.capterra.es/software/71249/firebird)



Esta base de datos relacional se ha usado en entornos de producción (bajo distintos nombres) desde 1981 y ofrece muchos estándares ANSI SQL. Firebird puede ejecutarse en Linux, Windows y varias plataformas Unix.

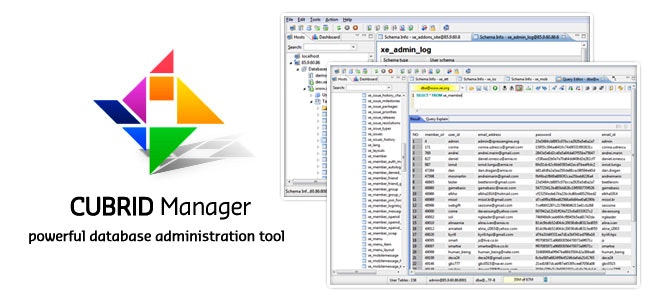
Ventajas

* Rastreo de API para supervisión en tiempo real
* Autenticación fiable en Windows
* Soporta cuatro arquitecturas:
  + SuperClassic, Classic, SuperServer y Embedded
* Distintas herramientas de desarrollo:
  + Herramientas comerciales: FIBPlus e IBObjects
* Opción de limpieza automática de la base de datos
* Notificaciones de eventos mediante alertas de base de datos y procedimientos almacenados
* Soporte gratuito gracias a la gran comunidad global de Firebird

Desventajas

* No incluye soporte de replicación integrada (solo como complemento).
* Carece de tablas temporales e integración con otros sistemas de base de datos.
* Carece de autenticación fiable en Windows en comparación con otros sitemas SO.

[CUBRID](https://www.capterra.es/software/105796/cubrid)



CUBRID es una fantástica solución gratuita y de código abierto especialmente optimizada para aplicaciones web. Es muy útil para servicios web complejos que procesan grandes cantidades de datos y generan multitud de solicitudes simultáneas. Esta solución está implementada en lenguaje C.

Ventajas

* Bloqueo de granularidad múltiple
* Copia de seguridad en línea
* Herramientas y drivers de interfaz gráfica de usuario para varios lenguajes de desarrollo:
  + JDBC, PHP, Python, Perl y Ruby
* Función de conmutación automática con servicio web en línea permanente
* Soporta fragmentación de BD nativas para escalabilidad horizontal y vertical
  + Los sistemas grandes dividen los datos en múltiples instancias de bases de datos.
* Replicación de bases de datos y coherencia en las transacciones

Desventajas

* No funciona con sistemas Apple
* No cuenta con depurador de script
* El manual solo se encuentra disponible en inglés y en coreano

**NoSQL**

**MongoDB**

[](https://www.grapheverywhere.com/wp-content/uploads/2019/06/instalando-mongodb-t1.png)

MongoDB logo

Desde las **Bases de datos documentales** una de las favorita de los desarrolladores en **MongoDB de 10Gen.**Sus inicios se remontan a finales de 2007, pero su liberación en licencia **opensource** fue en el año 2009. Este importante gestor de datos **NoSQL**almacena documentos en un formato muy parecido al **JSON** a alta velocidad.

Construido en **C++** tiene un rendimiento increíble que le permite ser muy flexible y potente. Es ideal para proyectos en los que se requiera alto nivel de **escalabilidad**. Tiene capacidad para recibir miles de lecturas por segundo sin pestañear. Múltiples gestores de contenido y juegos online se benefician de sus propiedades.

**Apache Cassandra**

[](https://www.grapheverywhere.com/wp-content/uploads/2019/06/cassandra-logo.gif)

Logotipo – Apache Cassandra

**ApacheCassandra**nace desde como un proyecto de google. Varios procesos han transcurrido desde esa época, hasta que en 2010 obtuvo su graduación como proyecto de alto nivel en **ApacheIncubator**.

Es una base de datos de tipo **clave-valor**. **Cassandra**es una **BBDD NoSQL**está diseñada para almacenar cantidades gigantescas de datos y realizar distribuciones a través de varios nodos. Esto permite que el almacenamiento de datos pueda estar repartido entre diversos servidores sin un solo punto de fallo. La gran mayoría de nosotros tiene una cercanía con esta **base de datos** ya que es una de las herramientas esenciales de **Facebook**.

**CouchDB**

**CouchDB** es una **base de datos NoSQL**que nace con aspiraciones bastante altas. Su principal desarrollador aspiraba a que **CouchDB**se convirtiera en la base de datos más usada en internet. En 2008 el proyecto pasa a formar parte de **ApacheIncoubator**.

La **BBDD CouchDB**llega a nosotros con la intención de facilitar la accesibilidad y compatibilidad web con diferentes tipos de dispositivos. Su primera versión estable llegó al público en el año 2010. En esta **BBDD**los datos se incluyen en formato **JSON**. La forma en la que los datos son organizados es según pares de valor clave.

**Redis**

Otro importante **motor de base de datos** de tipo **calve-valor** es Redis. Esta **BBDD NoSQL** de Código abierto es patrocinada y desarrollada por **RedisLabs**. Su diseño principal está basado en el almacenamiento de **tablas de hashes** aunque no es restrictiva sólo hacia este modelo. También tiene la posibilidad de ser utilizada como una BBDD persistente.

**Neo4j**



Logo de Neo4J

La **base de datos Neo4j**es desarrollada en **software** **libre** es del tipo **orientada a grafos**, construida con lenguaje **Java**. Esta **base de datos**tiene un funcionamiento transaccional que almacena en estructura de **grafos**. Su primera versión fue lanzada en 2010. Esta **BBDD**con diferentes tipos de **grafos**.

Las características principales que ofrece **Neo4J**a sus desarrolladores es un **alto desempeño,**si las consultas empiezan a crecer de forma exponencial el rendimiento de esta **BBDD**no disminuye. Es **Ágil, flexible y**altamente escalable. De sus usos y posibilidades de implementación te hablaremos con seguridad más adelante.

**Tipos de Bases de Datos NoSQL**

Ya conocimos algunos nombres famosos de las **Bases de Datos NoSQL**, pero es importante destacar que existe una **clasificación** que debemos conocer para entender cómo estas **BBDD** están construidas y nos pueden ser de utilidad. Veamos cuales son las principales clasificaciones:

[](https://www.grapheverywhere.com/wp-content/uploads/2019/06/cabinet-data-data-center-325229.jpg)

Photo by rawpixel.com from Pexels

**Bases de datos documentales**

Una base de datos documental u orientada a documentos, es un tipo de **BBDD NoSQL**que orienta su funcionamiento a datos alojados en **documentos** o **tiendas de documentos**. Estas bases de datos se utilizan para almacenar, recuperar y administrar datos semiestructurados.

Las bases de datos documentales almacenan cada uno de los registros y los datos asociados en un solo documento. Cada documento contiene **datos semiestructurados** que pueden ser consultados con diferentes herramientas de **análisis.**Estas bases de datos ofrecen mucha **flexibilidad**, **escritura rápida**, y **consultas rápidas**gracias a su gran capacidad de indexación. Entre las BBDD documentales más reconocidas encontramos la ya conocida por nosotros a **MongoDb, de 10gen**, y **CouchDB, de Apache**.

También gozan de un gran reconocimiento en el mundo de las Bases de datos NoSQL la [**RavenDB, de Hibernating**](https://ravendb.net/)**Rhinos, BaseX, djondb, eXist, SimpleDB, de Amazon, IBM Lotus Domino**y**Terrastore**

**Bases de datos en orientadas a grafos**

Las **bases de datos orientadas a grafos**son excelentes herramientas para trabajar datos complejos. Estas **BBDD**nos permite darle representar los datos en estructuras de **grafos**. Esto es de gran utilidad cuando los datos que vamos a procesar tienen altos niveles de interrelación.  Estas versátiles bases de datos, nos permite ejecutar consultas y almacenamiento de datos de cualquier característica sin siquiera preocuparnos por el volumen de los datos.

Gozan de un excepcional rendimiento para responder de forma eficiente al análisis y consulta de volúmenes gigantescos de datos. Ofrece también una diversidad de métodos analíticos y de consulta que la convierten en una de las opciones más flexibles en **NoSQL**. Es bastante frecuente conseguir la implementación de este modelo de **BBDD**en estructuras web de blogs. **Twitter**es uno de los casos más relevantes donde una BBDD de Grafos está relacionada.

Las **Bases de Datos orientadas a grafos**también tiene otras [funcionalidades](https://www.grapheverywhere.com/opendata-grafos-ayuntamiento-madrid/) que no te puedes perder. Entre las marcas más reconocidas está nuestra ya conocida **Neo4j**, pero esta no es la única. También están las conocidas **HyperGraphDB**, **InfoGrid**, **AllegroGraph, InfiniteGraph**, **Sones y DEX/Sparksee**.

**Bases de datos clave/valor**

Las **bases de datos clave valor** son modelos no relacionales que utilizan un método simple de almacenamiento de datos. Este tipo de **BBDD**toma los datos como un conjunto de pares “**clave-valor**” en los que las claves cumplen funciones de **identificadores únicos**.

Las claves o lo valores pueden ser cualquier tipo de datos. Pueden ser objetos simples o complejos. Estas **BBDD** son altamente divisibles y brindan grandes capacidades de escalabilidad horizontal.

Estos modelos de **BBDD** son uno de los preferidos de los clientes NoSQL, porque son simples en cuanto a funcionalidad y brindan alto rendimiento a la hora de ejecutar las lecturas y escrituras de datos. Ya hemos dado algunos ejemplos de **Bases de Datos Clave Valor** como **Cassandra** y **Redis**, pero es obligatorio expandir la lista cuando tenemos importantes BBDD como [**BigTable de Google**](https://cloud.google.com/bigtable/?hl=es), Dynamo de Amazon, Project Voldemort de LinkedIn, OracleNoSQL y Riak.

**Bases de datos multivalor**

Las bases de datos multivalor son sistemas interesantes que incorporan diferentes características multidimensionales  y NoSQL para la clasificación y  manejo de los datos. Estas **BBDD**comparten significativas similitudes con los modelos relacionales tradicionales. Ambos esquemas contienen tablas. Pero que esto no te engañe, las BBDD **multivalor** proporcionan un esquema de trabajo menos rígido.

Además de proporcionar mayor flexibilidad, los datos almacenados acá pueden contener listas de valores. Esto quiere decir que cualquier dato almacenado puede tener diversos valores asignados.

Estas bases de datos tienen un nivel de complejidad un poco más elevado debido a que incorporan reglas de normalización para su diseño. Entre las BBDD multivalor más conocidas podemos destacar [**Rocket D3 DBMS**](https://www.rocketsoftware.com/products/rocket-d3)**,  Rocket mvBase DBMS, Rocket U2 Universe, Rocket U2 Unidata, OpenQM,  Caché InterSystems, Reality, Jbase, OpenInsight, Extensible storage engine**

**Bases de datos orientadas a objetos**

Como bien lo indica su nombre, las **BBDD** de este tipo están conformadas por objetos. Estos objetos pueden ser de diferentes tipos, sobre los que se definen unas operaciones que determinan sus propiedades de interacción. Las **Bases de datos orientadas a objetos** han revivido el interés de los usuarios gracias a sus características principales.

Estas **BBDD**son altamente flexibles y admiten lenguajes de programación como **JAVA, Visual Basic**o **C++**.  Entre las más **Bases de Datos orientadas a objetos**más conocidas tenemos a **ObjectDB**, **Zope Object Database,**ZooDB, [**GemStone**](https://en.wikipedia.org/wiki/Gemstone_(database)), **Objectivity**y **Realm.io**

**Bases de datos tabulares**

Una **BBDD tabular** no es más que la estructuración de una BBDD en forma de tabla. Incorpora elementos en columnas y líneas. Cada una de las celdas genera intersecciones entre las columnas y las líneas. A estas intersecciones se le asignan una numeración única para establecer un orden eficiente de los datos. Están pensadas para grandes volúmenes de datos.

Estas tienen la capacidad de almacenar gran cantidad de datos dispersos. Entre las principales DDBB de este estilo podemos conseguir a **HBase de Apache** que es utilizada para soportar el servicio de mensajería de **Facebook**, también a BigTable de Google y la versión abierta llamada LevelDB y a Hypertable.

**Bases de datos de Arrays**

Las **Bases de datos arrays**sirven para trabar colecciones de datos conocidas como ***raster data***. Sitúan los datos en una cuadricula regular con más de dos dimensiones. Estas bases de datos se utilizan para representar simulaciones, sensores y datos estadísticos. Son capaces de manejar volúmenes de datos importantes ofreciendo una flexibilidad y escalabilidad.

Estas bases de datos son consideradas una generación tecnológica emergente. De las bases de datos más destacadas que trabajan este modelo podemos mencionar a Oracle que ha profundizado sus trabajos en ella y [**SciDB, de Paradigm4**](https://www.paradigm4.com/)

**Ventajas y desventajas de las Bases de datos NoSQL**

Una vez estudiadas y descritas los múltiples modelos de **Bases de Datos NoSQL** que tenemos a nuestra disposición, es casi obligatorio resumir sus ventajas y desventajas, para poder decidir en qué momento aprovecharlas. También describiremos algunas desventajas ciertas. No todo es color de rosa y las **BBDD relacionales**aún tienen un lugar importante en este mundo.

**Ventajas**

* **Alto rendimiento y baja exigencia**: un punto común de las BBDD NoSQL es que pueden ejecutarse en máquinas con recursos modestos. A diferencia de los sistemas **SQL**, la inversión que necesitamos hacer en estos casos es significativamente baja.
* **Escalabilidad horizontal**: para mejorar el rendimiento de estos sistemas no debemos inventar un mundo nuevo o una revolución total. Podemos incrementar o añadir nodos de forma horizontal para pisar a fondo el acelerador de rendimiento.
* **El volumen de datos no es problema**: se pueden manejar gran cantidad de datos sin dudarlo. No se generan cuellos de botella ni se corre el riesgo de ralentizar el sistema.
* **Flexibilidad**: como ya pudimos detallar en cada uno de sus tipos, estas BBDD NoSQL son extremadamente flexibles y podemos hacer que se ajusten a casi todo lo que necesitamos.

**Desventajas:**

* **Funciones de Fiabilidad:**esta si se quiere es una de las debilidades más grandes de los modelos **NoSQL**. Estas bases al no admitir funciones de fiabilidad lleva a los desarrolladores a implementar su propio código, lo que agrega mayor complejidad a los sistemas.
* **Aplicabilidad**: Esta desventaja está íntimamente relacionada con la anterior. La falta de inclusión de funciones de fiabilidad limita la aplicabilidad de estas BBDD a funciones delicadas como las del sector bancario, limitando mucho su potencial.
* **Incompatibilidad en consultas SQL:**en la mayoría de los casos las BBDD NoSQL son incompatibles a consultas SQL, lo que agrega una dificultad mayor ya que en caso de ser necesario se debe incluir una consulta manual, que puede hacer los procesos más lentos y complejos.

**Ejemplos y casos de uso de Bases de datos NoSQL**

Como vemos las BBDD NoSQL gozan de múltiples funciones, marcas, tipos y ventajas. Ahora, es momento de poner más en contexto y mostrar algunos ejemplos de su implementación.

A modo de ejemplo utilizaremos un **JSON** como el que conseguirás en **mongoDB**.

Veamos:

Supongamos que vamos a registrar diferentes personas en una colección perteneciente a una BBDD NoSQL con algunos campos especiales. Estos no necesariamente tienen que seguir un patrón específico como verás a continuación:

{  
Nombre: «José»,  
Apellidos: «Pérez Campo»,  
Edad: 35,  
Aficiones: [«vino»,»libros»,»ciclismo»],  
Amigos: [  
{  
Nombre:»María»,  
Edad:22  
},  
{  
Nombre:»Luis»,  
Edad:28  
}  
]  
}

Ahora bien, si queremos añadir a otros datos de otra persona con algunas características diferentes la podemos hacer sin mayor problema introduciendo lo siguiente:

{  Nombre: "Luis",

Estudios: "Marketing y Publicidad",

Amigos:12

}

En un modelo relacional o SQL clásico esto sería imposible de hacer. Esta es una de las tantas ventajas de las que te hemos hablado.