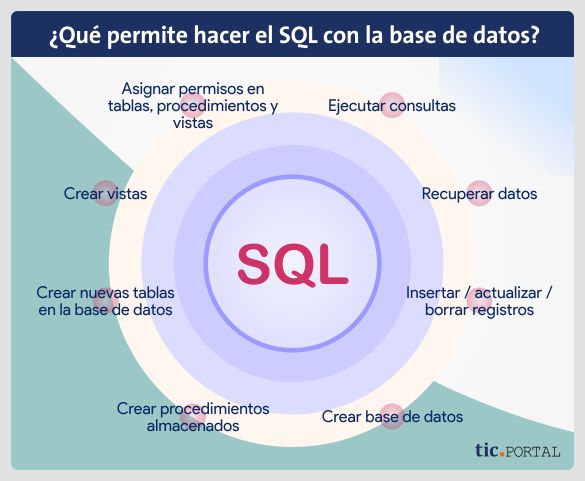
Base de Datos SQL

Una base de datos SQL es aquella **base de datos relacional** que esta **escrita en lenguaje SQL** (*Structured Query Language* o lenguaje de consulta estructurado), también pronunciado «sequel». Este lenguaje se considera el **lenguaje estándar para las bases de datos según el ANSI** (*American National Standards Institute* o Instituto Nacional Americano de Estándares en español), aunque no es el único que hay.



Una vez que se tiene una base de datos, hace falta un **sistema que sea capaz de crear y proporcionar acceso** ([acceso de distintos niveles](https://www.ticportal.es/glosario-tic/gestion-identidad-acceso)) a los datos de la [base de datos](https://www.ticportal.es/glosario-tic/base-datos-database). Este es el sistema de gestión de base de datos (**DBMS**, *Data Base Management System*). Algunos ejemplos de sistemas de bases de datos comunes son: [SAP HANA](https://www.ticportal.es/noticias/base-de-datos/sap-hana), Microsoft SQL Server y [Amazon RDS](https://www.ticportal.es/temas/cloud-computing/amazon-web-services/amazon-rds).

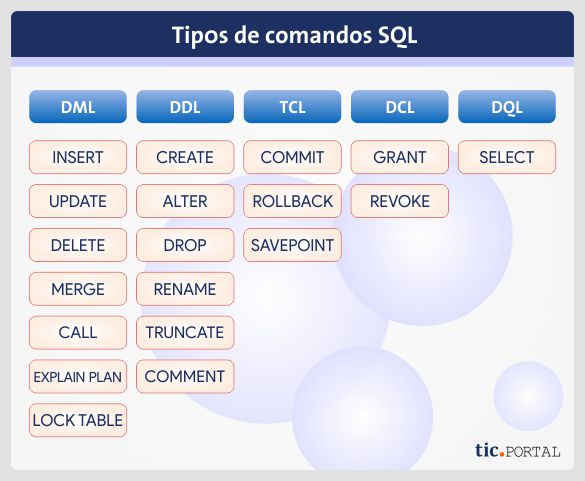
Para hacer una base de datos se necesitan casi únicamente los **comandos básicos de SQL** como: «Seleccionar», «Insertar», «Actualizar», «Eliminar», «Crear» y «Eliminar». No obstante, después **cada DBMS tiene una pequeña variación del SQL** con la cual puede agregarle funcionalidades, haciendo que el sistema de base de datos sea única. Esta situación se asemeja a la de comparar el SQL al español y, después, el lenguaje de cada DBMS al de las distintas comunidades autónomas.

**Nota**: es fácil sentirse perdido al ver tantas opciones de sistemas de gestión de base de datos. En la [Guía ERP 2020](https://www.ticportal.es/guias/guia-erp) encontrará una **comparación de distintas bases de datos SQL**.

¿Qué tipos de comandos hay dentro del SQL?

Para comprender lo que puede hacer una base de datos SQL hay que conocer los diferentes tipos de comandos que tiene. Los comandos del lenguaje SQL se dividen según su función en estos **5 tipos**:

* **DDL** (*Data Definition Language*): **definen el esquema** o estructura de la base de datos. Ejemplos: CREATE (crear); ALTER (alterar); DROP (eliminar objetos); RENAME (renombrar); TRUNCATE (quita todos los registros de una tabla, incluidos los espacios de los registros eliminados); COMMENT (comentar); entre otros.
* **DQL** (*Data Query Language*): sirven para hacer **consultas sobre los datos** en el esquema de objetos. Un objeto puede ser desde un resultado de búsqueda a una tabla. El propósito del comando es el de establecer una relación, basada en la consulta, dentro de la estructura de la base de datos, como la función de búsqueda. Ejemplo: SELECT (recuperar registros de la base de datos).
* **DML** (*Data Manipulation Language*): tratan la **manipulación de los datos** presentes en la base de datos. La mayoría de los comandos pertenecen a este tipo. Ejemplo: INSERT (insertar un objeto); DELETE (eliminar registros); UPDATE (actualizar); CALL; MERGE (3 en 1, inserta, elimina y actualiza); LOCK TABLE (bloquear tabla); EXPLAIN PLAN (determina el plan de acceso); …
* **DCL** (*Data Control Language*): se encargan de los **derechos, los permisos y otros controles** del sistema de la base de datos. Ejemplos: GRANT (proporcionar privilegios acceso a un usuario); REVOKE (revocar el derecho de accceso dado a un usuario); etc.
* **TCL** (*Transaction Control Language*): sirve para las transacciones con la base de datos. Es decir, con estos comandos se puede **llevar un control sobre otros comandos y cómo afectan** a la base de datos. Ejemplos: COMMIT (llevar a cabo una transacción); ROLLBACK (revertir una transacción en caso de que ocurra algún error); SAVEPOINT (establecer un punto de rescate dentro de una transacción.).



Diferencias entre una base de datos SQL y una NOSQL

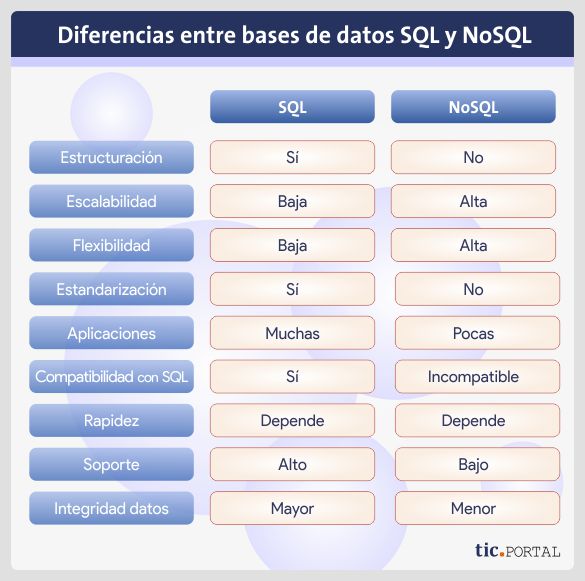
A la base de datos tradicional, la relacional SQL, le ha salido una competidora, la NoSQL (*Not Only SQL*, no solo SQL). La **popularidad de la base de datos NoSQL** se debe a grandes empresas como Facebook, Twitter, Google o Amazon. Estas empresas la usan para proporcionar **grandes volúmenes de información** en el menor tiempo posible, aunque para ellos sacrifican la consistencia de los datos. Las **redes sociales usan la base de datos NoSQL** porque así pueden procesar grandes cantidades de información creando la estructura de grafos (tipo abstracto de datos) con el que se establecen relaciones entre las personas.

En cuanto a la rapidez, una base de datos será **más rápido** que la otra dependiendo del contexto. Por ejemplo, las bases de datos **SQL serán más rápidas** que las NoSQL cuando se trata de uniones, consultas, actualizaciones, etc. En cambio, las bases de datos **NoSQL son más rápidas** que las SQL cuando se trata de realizar operaciones de lectura o escritura en una sola entidad de datos. Esto se debe a que las bases de datos NoSQL están diseñadas especialmente para datos desestructurados, los cuales pueden estar orientados a: documentos (JSON); columnas; gráficos; etc. Así, una entidad particular de datos está almacenada de forma conjunta y no está en particiones.

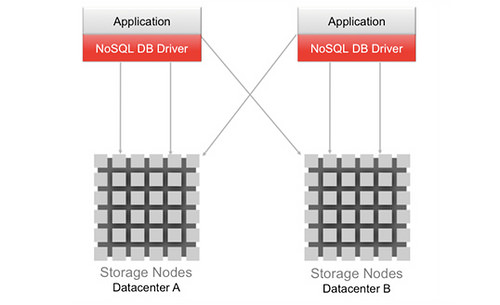
Como se ha mencionado, la base de datos NoSQL no está estructurada, al igual que tampoco está centralizada, permitiendo una **mayor escalabilidad**. Además, este tipo de base de datos no necesita tantos recursos para ejecutarse ya que es **abierta** y, por lo tanto, **flexible**.

No obstante, al no tener un lenguaje estandarizado, el **NoSQL presenta una serie de inconvenientes** como que no haya un gran soporte, ni muchas aplicaciones (plug-ins), menor fiabilidad y coherencia en sus datos. Asimismo, a menudo hay que conectar varias bases de datos y la NoSQL es incompatible con SQL.

Por tanto, a la base de datos **NoSQL todavía le queda un largo camino** que recorrer. Sin embargo, ya es útil para muchas empresas. La base de datos **NoSQL es rentable** para aquellas empresas en las que el volumen de datos crece muy rápidamente en momentos puntuales; cuando necesita una gran escalabilidad; o, cuando el esquema de la base de datos no es homogéneo. Algunos ejemplos de **DBMS de NoSQL** son: MongoDB, Redis y Apache CouchDB.



Base de Datos NoSQL



***Por Andrés Araujo, Principal Technical Sales Consultant***

**Cuando hablamos de entornos de base de datos o de desarrollo de aplicaciones web, cada vez es más frecuente el uso del término NoSQL. Pero, ¿qué es una base de datos NoSQL?**

Pese a la no existencia de una definición formal, cuando hablamos de base datos NoSQL, también conocidas como “No sólo SQL”, nos referimos a una amplia clase de sistemas de gestión de datos (mecanismos para el almacenamiento y recuperación de datos) que difieren, en aspectos importantes, del modelo clásico de relaciones entre entidades (o tablas) existente en los sistemas de gestión bases de datos relacionales, siendo el más destacado el que no usan SQL como lenguaje principal de consulta.

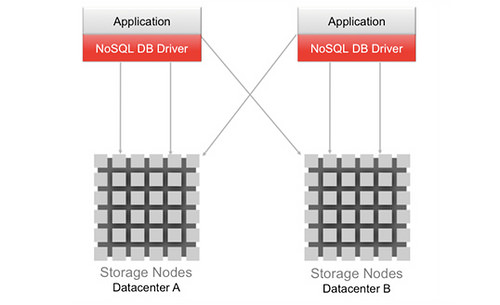
Aunque son conocidas desde la década de los 60 del pasado siglo, su auge actual viene determinado por el uso que, de estos sistemas han hecho las principales compañías de internet como Amazon, Google, Twitter y Facebook. Estas compañías tenían que enfrentarse a nuevos desafíos en el tratamiento de los datos motivados por el enorme crecimiento de la Web donde se requería dar respuesta a la necesidad de proporcionar información procesada a partir de grandes volúmenes de datos con unas estructuras horizontales, más o menos, similares y con aplicaciones web que debían dar respuesta a las peticiones de un número elevado e indeterminado de usuarios en el menor tiempo posible. Estas compañías se dieron cuenta de que el rendimiento y sus necesidades de tiempo real eran más importantes que la consistencia de los datos, aspecto este último al que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

Las características comunes entre todas las implementaciones de bases de datos NoSQL suelen ser las siguientes:

* **Consistencia Eventual**: A diferencia de las bases de datos relacionales tradicionales, en la mayoría de sistemas NoSQL, no se implementan mecanismos rígidos de consistencia que garanticen que cualquier cambio llevado a cabo en el sistema distribuido sea visto, al mismo tiempo, por todos los nodos y asegurando, también, la no violación de posibles restricciones de integridad de los datos u otras reglas definidas. En su lugar y para obtener un mayor rendimiento, se ofrece el concepto de “consistencia eventual”, en el que los cambios realizados “con el tiempo” serán propagados a todos los nodos por lo que, una consulta podría no devolver los últimos datos disponibles o proporcionar datos inexactos, problema conocido como *lecturas sucias* u *obsoletas*.  
  Asimismo, en algunos sistemas NoSQL se pueden presentar perdidas de datos en escritura. Esto se conoce también como BASE (*Basically Available Soft-state Eventual Consistency*), en contraposición a ACID (*Atomicity, Consistency, Isolation, Durability*), su analogía en las bases de datos relacionales.
* **Flexibilidad en el esquema:**En la mayoría de base de datos NoSQL, los esquemas de datos son dinámicos; es decir, a diferencia de las bases de datos relacionales en las que, la escritura de los datos debe adaptarse a unas estructuras(o tablas, compuestas a su vez por filas y columnas) y tipos de datos pre-definidos, en los sistemas NoSQL, cada registro (o documento, como se les suele llamar en estos casos) puede contener una información con diferente forma cada vez, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras complejas de datos en un sólo documento, como por ejemplo almacenar la información sobre una publicación de un blog (título, cuerpo de texto, autor, etc) junto a los comentarios y etiquetas vertidos sobre el mismo, todo en un único registro.
* **Escalabilidad horizontal**: Por escalabilidad horizontal se entiende la posibilidad de incrementar el rendimiento del sistema añadiendo, simplemente, más nodos (servidores) e indicando al sistema cuáles son los nodos disponibles.
* **Estructura distribuida**: Generalmente los datos se distribuyen, entre los diferentes nodos que componen el sistema. Hay dos estilos de distribución de datos:
  + **Particionado (ó *Sharding*)**: El particionado distribuye los datos entre múltiples servidores de forma que, cada servidor, actúe como única fuente de un subconjunto de datos. Normalmente, a la hora de realizar esta distribución, se utilizan mecanismos de tablas de *hash* distribuidas (DHT).
  + **Réplica:** La réplica copia los datos entre múltiples servidores, de forma que cada bit de datos pueda ser encontrado en múltiples lugares. Esta réplica puede realizarse de dos maneras:
    - Réplica maestro-esclavo en la que un servidor gestiona la escritura de la copia autorizada mientras que los esclavos se sincronizan con este servidor maestro y sólo gestionan las lecturas.
    - Réplica *peer-to-peer* en la que se permiten escrituras a cualquier nodo y ellos se coordinan entre sí para sincronizar sus copias de los datos
* **Tolerancia a fallos y Redundancia**: Pese a lo que cualquiera pueda pensar cuando se habla de NoSQL, no todas las tecnologías existentes bajo este paraguas usan el mismo modelo de datos ya que, al ser sistemas altamente especializados, la idoneidad particular de una base de datos NoSQL dependerá del problema a resolver. Así a todo, podemos agrupar los diferentes modelos de datos usados en sistemas NoSQL en cuatro grandes categorías:​
  1. **Base de datos de Documentos**: Este tipo de base de datos almacena la información como un documento, usando para habitualmente para ello una estructura simple como JSON, BSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenidodel documento. Son las bases de datos NoSQL más versátiles.
  2. **Almacenamiento Clave-Valor:** Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una clave única, lo que permite la recuperación de la información de forma muy rápida, información que suele almacenarse como un objeto binario. Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras.
  3. **Bases de datos de grafos:**Usadas para aquellos datos cuyas relaciones se pueden representar adecuadamente mediante un grafo. Los datos se almacenan en estructuras grafo con nodos (entidades), propiedades (información entre entidades) y líneas (conexiones entre las entidades).
  4. **Base de datos Columnar (o Columna ancha)**: En vez de“tablas”, en las bases de datos de columna tenemos familias de columnas que, son los contenedores de las filas. A diferencia de los RDBMS, no necesita conocer de antemano todas las columnas, cada fila no tiene por qué tener el mismo número de columnas. Este tipo de bases de datos se adecuan mejor a operaciones analíticas sobre grandes conjuntos de datos.

​Pese a todas las opciones proporcionadas por el auge de las bases de datos NoSQL, esto no significa la desaparición de las bases de datos de RDBMS ya que son tecnologías complementarias. Estamos entrando en una era de persistencia políglota, una técnica que utiliza diferentes tecnologías de almacenamiento de datos para manejar las diversas necesidades de almacenamiento de datos.

Dentro de los diferentes sistemas de gestión de basede datos, y más concretamente en este ámbito, Oracle ofrece una base de datosNoSQL, con características empresariales, denominada Oracle NoSQL; basada, al  
igual que otras bases de datos NoSQL, en el motor de Oracle Berkeley DB, estaes una base de datos NoSQL distribuida y escalable, diseñada para proporcionar una gestión de datos fiable, flexible y altamente disponible a través de un conjunto configurable de nodos de almacenamiento, junto con una API de fácil uso e integración con el resto de productos Oracle.



Pese a ser pertenecer a la categoría de almacenamientos Clave-Valor, la base de datos NoSQL de Oracle, permite modelar  los datos como si fuesen tablas en forma similar a como lo haría una base dedatos relacional, almacenar documentos JSON o, simplemente, pares de clave y valor. Al igual que otras bases de datos NoSQL, Oracle NoSQL, es un sistema particionado(donde no se comparte nada), que distribuye los datos de manera uniforme entre las múltiples particiones que lo conforman, en base al valor hash de la clave primaria. Dentro de cada partición, los nodos de almacenamiento se replican con el fin de garantizar una alta disponibilidad, una rápida conmutación en el caso de un fallo de nodo y realizar un balanceo óptimo de la carga que representen las consultas. La base de datos NoSQL de Oracle base de datos proporciona drivers Java, C, Python y Node.js, así como, una API REST para simplificar el desarrollo de aplicaciones web.

Así mismo, se integra con una amplia variedad de aplicaciones Oracle y de código abierto con el fin de simplificar y agilizar el desarrollo y despliegue de aplicaciones modernas en entornos big data.