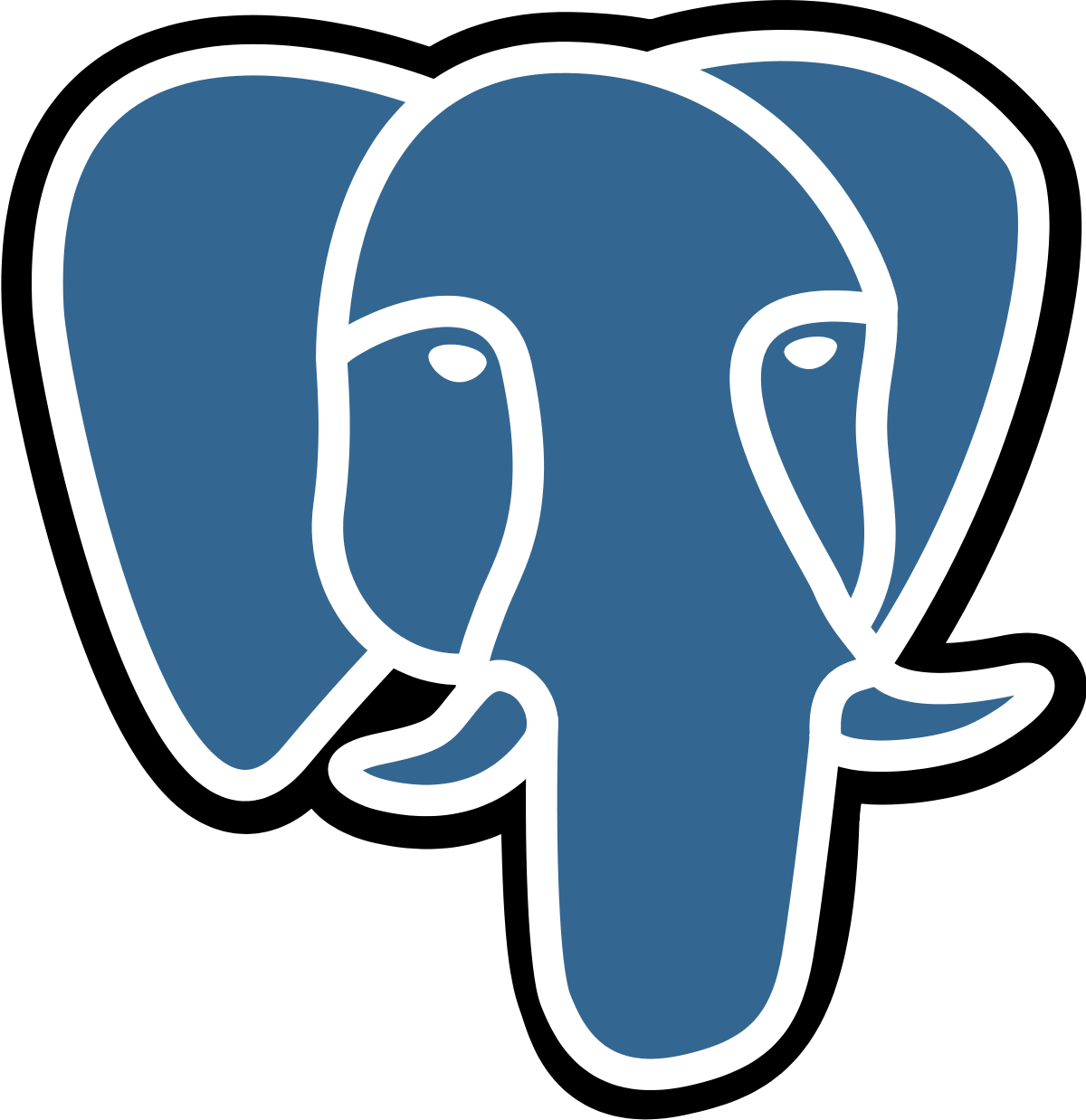
|  |  |  |
| --- | --- | --- |
|  |  | sQL distribuido  Juan Casado Ballesteros  Gino Cocolo Rodríguez |



# Problema Existente

Con el comienzo de la computación cloud comienza a haber un problema de rendimiento para manejar los datos. El número de usuarios que utilizan los sistemas crece exponencialmente y las arquitecturas tradicionales se quedan obsoletas.

Las bases de datos SQL deben actualizarse convirtiéndose en sistemas distribuidos que puedan ser capaces de soportar la demanda.

Los principios ACID por los que se rigen las bases de datos hacen que al convertirse en sistemas distribuidos solo puedan garantizar la disponibilidad y la consistencia de los datos.

* **Atomicidad**: todo o nada.
* **Consistencia**: coherencia de los datos.
* **Aislamiento**: serialización de transacciones.
* **Durabilidad**: los cambios son permanentes.

En este punto surgirán otros tipos de bases de datos NoSQL que se centrará en garantizar las otras dos combinaciones de servicios.

* Consistencia + Tolerancia a las particiones
* Disponibilidad + Tolerancia a las particiones

Dependiendo de aquello que necesitemos deberemos elegir un tipo de sistema u otro.

Pero aun así, estas soluciones basadas en NoSQL no podían ofrecer, por cómo han sido diseñadas, esa misma consistencia que aseguraban los aquellas bases de datos que seguían como filosofía los principios ACID. Eso condujo a la aparición de un nuevo diseño.

# ¿Qué es?

SQL distribuido es el nuevo diseño que surgió en esa situación de obsolescencia del SQL tradicional en algunos de los nuevos escenarios de trabajo que habían aparecido. Es la respuesta a la necesidad de escalar las bases de datos tradicionales.

Funciona utilizando una serie de bases de datos diseminadas en distintos emplazamientos en una misma red, interrelacionadas entre sí, con un mismo gestor de base de datos que maneja todas las consultas y operaciones, dando la sensación al usuario que está tratando con una base de datos relacional tradicional.

Esto permite que a la hora de pedir datos, aunque provengan de diferentes equipos de la red, no haya que preocuparse por eso. Ni de dónde ni cuantas replicaciones y copias hay. Ni de si los datos han sido fragmentados.

Hay dos formas mediante las que lograr esta distribución:

* Base de datos homogénea: Todas las distintas bases de datos trabajan de la misma forma. Comparten sistema operativo y estructuras de datos, así como gestor. También podemos diferenciar dentro de este tipo las que son autónomas, en las que cada base de datos es independiente y funciona por su cuenta, integrados en una aplicación que se encarga de controlarlas y compartiendo los cambios en los datos mediante el envío de mensajes; y las no autónomas, en las que la información se distribuye entre los nodos y un gestor de base de datos maestro coordina los cambios.
* Base de datos heterogénea: En este tipo, cada sitio puede contar con distintos esquemas, software, gestor… Esto hace que el proceso a la hora de consultar datos sea complejo, al igual que el procesado de las transacciones. Al no tener que ser consciente un nodo del resto de sus compañeros, esto puede ocasionar una bja cooperación cuando se intenta resolver una consulta o petición por parte del usuario. A su vez se dividen en federadas, en las que cada sistema es independiente en naturaleza e integrados juntos funcionando como uno solo; y no federadas, que incluyen un módulo central con la tarea de coordinar a través del cual se produce el acceso a las bases de datos.

También encontramos diferencias de diseño a seguir para la distribución de los datos:

* Sin replicación y sin fragmentación: Distintas tablas se guardan en distintos lugares. Generalmente, el criterio a seguir a la hora de almacenar, es almacenarlo geográficamente próximo a aquellos usuarios que más lo usen. Este diseño es utilizado cuando las consultas tienen una baja necesidad de información conjunta por varias tablas, y permite reducir costes de comunicación.
* Totalmente replicada: Cada sitio guarda una copia de la base de datos en su totaldad. Esto tiene como ventaja su rapidez al no tener que comunicarse con otros nodos a la hora de servir datos de las consultas. Pero como desventaja el gran coste que conlleva realizar una actualización de los datos, al tener que realizarse en todos los nodos. Es obvio que este diseño en sistemas con un gran volúmen de consultas pero con baja necesidad de actualización es ideal.
* Parcialmente replicada: Copias de las tablas o de parte de las tablas se guardan en distintos sitios. La decisión de dónde almacenar qué viene determinada por la frecuencia de acceso a los datos. Y el número de copias a guardar dependerá también de la frecuencia de consulta y el usuario que las realiza.
* Fragmentado: Las tablas se dividen en varios fragmentos y son guardados en distintos sitios. Este diseño no cuenta con datos duplicados, es decir, elimina la redundancia. También procura incrementar el paralelismo de las operaciones. La fragmentación puede ser horizontalmente, es decir, una tabla es dividida en dos o más subsets cada uno conteniendo todas las columnas, pero parte de las filas, o verticalmente, dividiendo esos subsets para que contengan parte de las columnas y todas las filas.

Independientemente de la naturaleza, todas comparten el tener un solo gestor con el que interactuar por parte del cliente. Esto ofrece la posibilidad de trabajar como hasta ahora, con todas las ventajas

# Solución

# Demostración

<https://github.com/bitnami/bitnami-docker-pgpool>

<https://portworx.com/ha-postgresql-kubernetes/>

biblio

<https://www.cockroachlabs.com/blog/what-is-distributed-sql/>

<https://blog.yugabyte.com/what-is-distributed-sql/>

<https://www.geeksforgeeks.org/advantages-of-distributed-database/>

<https://www.geeksforgeeks.org/distributed-database-system/>

<https://www.geeksforgeeks.org/functions-of-distributed-database-system/>

[https://www.tutorialspoint.com/distributed\_dbms/distributed\_dbms\_database\_environments.htm](https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_database_environments.html)