Engineer

Base de datos

almacén de datos organizado para que lo consulten programas

Qué es

Una base de datos es un conjunto de datos conservados en una estructura fija, con reglas sobre lo que puede introducirse en ella, para que el software pueda encontrar, cambiar y comprobar su contenido; la misma expresión, a menudo abreviada como DB, también se utiliza de manera imprecisa para el programa que los conserva, cuyo nombre exacto es sistema de gestión de bases de datos o DBMS. El glosario del NIST recoge dos definiciones, y la más amplia describe un repositorio de información o datos «que puede ser o no un sistema tradicional de base de datos relacional».

La expresión cumple tres funciones, y una conversación no avanza cuando dos personas se refieren a cosas distintas. Designa los datos: los clientes, pedidos y lecturas de sensores que conserva una organización. Designa el motor que los almacena: PostgreSQL, SQL Server, SQLite. Y dentro de un servidor designa un contenedor entre varios. El tutorial de PostgreSQL establece los niveles: las filas con columnas con nombre y tipo forman tablas, las tablas se agrupan en bases de datos y las bases de datos que una instancia de servidor gestiona conjuntamente forman un clúster de bases de datos. SQL Server establece la misma distinción entre una instancia y las bases de datos que aloja. «La base de datos va lenta» puede referirse a una tabla, un contenedor o una máquina completa.

La mayoría de los motores de uso cotidiano son relacionales: guardan los datos como relaciones, el término matemático para las tablas, y aceptan preguntas en SQL. El tutorial de PostgreSQL menciona otras organizaciones. Los archivos y directorios de un sistema de tipo Unix forman una base de datos jerárquica, y las bases de datos orientadas a objetos llegaron después. Redis se define como un servidor de estructuras de datos: en lugar de tablas ofrece cadenas, hashes, listas, conjuntos y otros tipos nativos, utilizados para caché, colas y procesamiento de eventos.

Los motores también difieren en dónde se ejecutan. Un motor cliente/servidor es un proceso de larga duración al que muchos programas acceden por la red. Un motor integrado es una biblioteca dentro de un programa, y SQLite guarda una base de datos completa, incluidas tablas, índices, disparadores y vistas, en un archivo corriente; el proyecto afirma que compite con `fopen()` más que con las bases de datos de servidor. La diferencia se aprecia en cómo se conecta un programa:

```
# client/server: connect to a service that is already running
psql "host=db.internal dbname=shop user=app"

# embedded: the program opens a file and the engine runs inside it
sqlite3 shop.db
```

Una tercera familia se construye para el análisis. BigQuery guarda cada columna por separado, mientras que muchos sistemas tradicionales conservan juntos en disco los campos de una fila. El almacenamiento por filas resulta adecuado para buscar un registro; el almacenamiento por columnas, para sumar un campo a lo largo de millones de filas, porque la consulta lee esa columna y deja las demás sin tocar.

Utilice el concepto en una decisión

Aplique «¿Qué es una base de datos?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Una estructura que refleja cómo funciona el negocio para poder responder también a las preguntas que surjan más adelante.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Diseño y operación de 6 flujos ETL/ELT — 6 flujos ETL en tres motores de bases de datos

Diseño y operación de 6 flujos ETL/ELT Póngase en contacto →

Por qué importa

Los archivos también contienen datos. Una base de datos añade garantías sobre esos datos para el momento en que muchos programas los utilizan a la vez.

La primera garantía es una visión coherente con uso concurrente. PostgreSQL da a cada sentencia una instantánea de los datos tal como estaban en un momento, un diseño llamado control de concurrencia multiversión o MVCC, de modo que una lectura nunca espera a una escritura y una escritura nunca espera a una lectura. SQLite admite cualquier número de lectores a la vez y un escritor en cada

instante, mientras los demás escritores esperan su turno. Ambos se adaptan a distintos números de escritores, y ese número determina buena parte de la elección.

La segunda garantía es que los datos conservan su estructura. Un esquema establece qué columnas tiene una tabla y qué tipo contiene cada una; las claves y restricciones establecen qué filas pueden existir. Cuando la base de datos aplica esas reglas, todos los programas que escriben en ella las obedecen. Cuando las reglas solo están en el código de la aplicación, el primer programa que se las salta deja filas frente a las que todas las consultas posteriores tienen que protegerse.

La tercera es un lenguaje compartido, con acentos. SQL es el estándar ISO/IEC 9075, revisado por última vez como SQL:2023; aun así, el apéndice de conformidad de PostgreSQL señala que ningún sistema actual afirma cumplir íntegramente su núcleo, Core. Un esquema y sus consultas pertenecen a su motor, y trasladarlos a otro es una migración con sus propias pruebas.

La propia documentación de SQLite convierte la elección de motor en una breve lista de comprobación: si los datos están al otro lado de una red respecto al código que emite el SQL, si muchos procesos escriben en el mismo instante o si el contenido se acerca al orden de los terabytes, cualquiera de esas condiciones apunta a cliente/servidor; en caso contrario, SQLite es la respuesta más sencilla. Los informes a gran escala añaden una pregunta más: si las lecturas intensivas deben ejecutarse sobre una copia analítica separada, alimentada por ETL, para que un informe largo nunca compita con las consultas cuyos resultados esperan los clientes.

Cómo se aplica en la práctica

En el extremo de mayor escala, desde un puesto de ingeniería de bases de datos en una empresa tecnológica en rápido crecimiento se crearon, gestionaron y optimizaron 100+ bases de datos en PostgreSQL, Microsoft SQL Server y Google BigQuery, la mayoría con datos GIS y series temporales, como lecturas de sensores, registros y métricas en tiempo real. El trabajo produjo esquemas normalizados y desnormalizados para PostgreSQL y SQL Server, con indexación espacial (PostGIS en el caso de PostgreSQL) y particionamiento temporal, mientras que BigQuery contenía grandes volúmenes de series temporales en tablas con particionamiento temporal y agrupación (clustering). Una vez que esas

bases de datos contaron con índices, vistas materializadas y caché, las consultas GIS y de series temporales se respondían en un 40–60 % menos de tiempo, y la monitorización automatizada redujo el tiempo de inactividad un 50 %.

En un sistema más pequeño, en una etapa documentada de su desarrollo, un único archivo SQLite contenía los datos en los que se basaban un CV, una hoja de referencias, una cartera de trabajos y una carta de presentación en tres idiomas. Esa base de datos tenía veintitrés tablas, y un comando las construía en diecinueve pasos ordenados, cualquiera de los cuales podía repetirse de forma segura. Se generaban seis formatos de salida a partir de la base de datos mediante 41 módulos Python que sumaban 10.580 líneas. Un documento dirigido a un tipo de lector era «una consulta en lugar de una copia»: un perfil seleccionaba qué logros aparecían y en qué orden. La contrapartida era que entonces un documento solo podía proceder de esa base de datos y de ningún otro lugar. Estas cifras de implementación describen aquella etapa registrada, no el repositorio actual ni su cobertura de idiomas.

Entre esos tamaños se encuentra la pregunta a la que llega toda base de datos con mucha actividad, y una nota sobre optimizar antes de fragmentar los datos entre servidores responde con un orden de trabajo: observar, ajustar, sacar las lecturas de la instancia principal, particionar las tablas alrededor del punto de mayor carga y fragmentar entre servidores solo cuando una máquina esté llena de trabajo que merezca la pena. Observar significa preguntar al propio PostgreSQL: `pg_stat_statements` para las sentencias que más cuestan en total, y después `EXPLAIN (ANALYZE, BUFFERS)` para ver qué hace la más costosa.

Dónde surgen los problemas

La mayoría de los problemas de una base de datos empiezan en una decisión temprana, cuando había pocos datos y parecía que la decisión no tenía coste.

Dos copias de un hecho dejan de coincidir. El generador de CV anterior existe porque los datos conservados como cuatro documentos se corrigen cuatro veces, hasta que dos de ellos discrepan y nada indica cuál está actualizado. Lo mismo ocurre con una tabla exportada a una hoja de cálculo y editada allí, o con un segundo servicio con su propia lista de clientes. Dé a cada dato un único almacenamiento responsable de él y haga que todas las demás copias indiquen de dónde procede y de cuándo es. La caché de la nota sigue estrictamente esta regla: puede decir a un comprador que un artículo está totalmente reservado, y todas las respuestas afirmativas siguen viniendo de la base de datos.

Llegan más máquinas antes de que nadie lea los planes. Dividir los datos entre servidores reparte las sentencias lentas entre más hardware sin abaratar ninguna, y añade un coste permanente a todo lo demás: como enumera la nota, cada migración, copia de seguridad, restauración e informe se ejecuta una vez por fragmento, y una unión entre fragmentos pasa a ser código de aplicación. Lea primero `pg_stat_statements` y los planes, y conserve lo que muestran como referencia inicial.

Una lectura llega a una copia retrasada. Una réplica en streaming asume parte de las lecturas de la instancia principal y va algo por detrás de ella. Un usuario que acaba de escribir algo y después lo lee de la réplica puede ver el valor anterior. Lea las escrituras recientes del propio usuario desde la instancia principal.

Los informes y los clientes comparten un motor. Un informe que recorre dos años de historial ocupa memoria y disco que necesitan las consultas de la propia tienda, y las personas que compran lo notan. Traslade el análisis intensivo a una copia en un almacén de datos y reserve la base de datos operativa para las muchas lecturas y escrituras pequeñas a las que da servicio.

Lecturas adicionales

- Glosario del NIST: base de datos — dos definiciones breves de publicaciones del NIST, una de las cuales admite bases de datos que no son relacionales.
- Tutorial de PostgreSQL: conceptos — tablas, filas, columnas, bases de datos y clúster de bases de datos, en una página.
- PostgreSQL: introducción a MVCC — cómo permiten las instantáneas que lectores y escritores trabajen sin bloquearse entre sí.
- PostgreSQL: conformidad con SQL — las partes y versiones de ISO/IEC 9075 y lo que significa cumplir el núcleo Core.
- Usos apropiados de SQLite — dónde encaja un motor integrado, dónde encaja mejor uno cliente/servidor y una lista de comprobación para decidir.
- Descripción general del almacenamiento de BigQuery — almacenamiento orientado a filas y a columnas, y por qué el análisis favorece al segundo.

Glosario

Elija según sus restricciones reales

Traiga un conjunto de datos representativo, las preguntas que debe responder y el cuello de botella actual. Podemos acordar cómo medir la calidad, la capacidad de procesamiento y la recuperación.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una base de datos?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/database/