Engineer

SQL

Lenguaje que entienden las bases de datos relacionales

Qué es

SQL es un lenguaje declarativo para definir, modificar y leer datos almacenados en tablas relacionales: se escribe el resultado que se quiere obtener y la base de datos decide cómo producirlo. Las siglas suelen desarrollarse como Structured Query Language y pronunciarse letra a letra o como «sequel», mientras que el estándar internacional que lo define, ISO/IEC 9075, lo llama simplemente Database Language SQL.

La palabra clave es «declarativo». La siguiente sentencia indica una tabla, una condición, una agrupación y un orden. No dice nada sobre índices, bucles ni sobre qué filas hay que leer primero:

```
SELECT country, count(*) AS signups
FROM accounts
WHERE created_at >= now() - interval '30 days'
GROUP BY country
ORDER BY signups DESC;
```

En PostgreSQL, ese texto pasa por cuatro etapas. Un analizador comprueba la sintaxis y construye un árbol; una fase de reescritura expande las vistas que mencione la sentencia; un planificador compara las posibles formas de ejecutarla, como un recorrido secuencial frente al uso de un índice; y, por último, un ejecutor obtiene las filas. La misma sentencia puede tardar un milisegundo o un minuto según lo que encuentre el planificador. Por eso leer un plan forma parte de escribir SQL.

El lenguaje tiene partes que reciben nombres distintos. La definición de datos crea y modifica tablas, esquemas e índices. La manipulación de datos inserta, actualiza, elimina y fusiona filas. Las consultas las leen mediante SELECT. Los

permisos y las transacciones determinan quién puede hacer qué y qué cambios se aplican juntos. Todo viaja como texto por una única conexión, de modo que SQL es el terreno común entre una aplicación, una herramienta de informes y una persona ante un terminal.

El estándar se ha revisado muchas veces; la revisión más reciente es SQL:2023, y cada motor implementa su propio subconjunto. El apéndice de conformidad de PostgreSQL indica que cumple al menos 170 de las 177 características obligatorias de Core y que, en el momento de redactarlo, ningún sistema de bases de datos afirmaba cumplir Core por completo. Además, cada motor añade su dialecto: Microsoft SQL Server utiliza Transact-SQL, BigQuery utiliza GoogleSQL y PostgreSQL añade PL/pgSQL para el código procedimental que se ejecuta dentro de la base de datos.

Utilice el concepto en una decisión

Aplique «¿Qué es SQL?» 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

La decisión que SQL plantea a un equipo es dónde realizar el trabajo con los datos: dentro de la base de datos, en el lenguaje para cuya ejecución se diseñó, o en código de aplicación que obtiene las filas y las procesa por su cuenta.

Una sentencia basada en conjuntos se ejecuta junto a los datos. Filtrar un millón de filas hasta quedarse con cuarenta dentro de la base de datos envía cuarenta filas por la red. Filtrarlas en la aplicación envía un millón y descarta la mayoría. Las operaciones JOIN, las agregaciones y la eliminación de duplicados siguen la misma aritmética. Trasladar ese trabajo a SQL suele ser la mejora de velocidad menos costosa para un sistema lento, porque se ejecuta en la base de datos que ya existe.

La responsabilidad sobre el código es el contrapeso. La lógica escrita en SQL vive en migraciones y funciones que necesitan las mismas revisiones, pruebas y control de versiones que cualquier otro código. Hay menos profesionales que

lean una función larga de PL/pgSQL con la misma soltura que Go o Python. Un reparto viable lleva a la base de datos lo que necesita ver muchas filas y mantiene en la aplicación las reglas que cambian con el negocio, donde es más fácil probarlas de forma aislada.

La portabilidad es la segunda contrapartida. El núcleo del lenguaje, SELECT con operaciones JOIN, GROUP BY y ORDER BY, se traslada de un motor a otro con pocos cambios. Las particularidades de cada dialecto se trasladan peor: la aritmética de fechas, los upserts, la paginación, las funciones JSON y el código procedimental difieren, y la migración entre bases de datos consume ahí la mayor parte del esfuerzo. Elegir un motor también significa elegir un dialecto; cuanta más lógica dependa de él, más costará una migración posterior.

La contratación es el tercer aspecto. SQL es una de las pocas competencias que comparten analistas, desarrolladores de backend, especialistas en ingeniería de datos y administradores de bases de datos. Por tanto, cualquiera de ellos puede leer y revisar una capa de datos escrita en SQL puro. Un ORM oculta el SQL hasta que una página se vuelve lenta y, llegado ese punto, alguien sigue teniendo que leer lo que ha generado.

Cómo se aplica en la práctica

En una plataforma, las funciones de PostgreSQL ensamblan cada respuesta de la API como un documento JSON completo y el manejador de Go lo reenvía como `json.RawMessage`, sin descodificarlo ni volverlo a codificar. La arquitectura de paso directo de JSON hace que dar forma a la respuesta sea una responsabilidad del SQL, allí donde ya están los datos. Una pequeña utilidad llamada QueryJSON convirtió ese patrón en el más sencillo de seguir, y desaparecieron los DTO, los mapeadores y las estructuras de respuesta que suele acumular una API por capas. El código de los manejadores se acortó mucho, y un cambio en lo que devuelve una función llega al cliente sin que nadie tenga que editar un manejador.

El mismo lenguaje tiene otro aspecto en una canalización de datos. En seis canalizaciones ETL/ELT que abarcaban BigQuery, MSSQL y PostgreSQL, los archivos en bruto se cargaban en esquemas de preparación con bcp, psql y SSIS, y la limpieza, la normalización y el enriquecimiento se escribían como procedimientos PL/pgSQL y T-SQL. Tres motores, cada uno con su dialecto, dentro de un solo trabajo: la cuestión de la portabilidad anterior, vista desde dentro. La página documenta el resultado: al menos tres horas de trabajo

manual reducidas a menos de 20 minutos de principio a fin, actualización diaria de los datos en lugar de semanal y API de Python que los consultaban un 30 % más rápidas.

Dónde surgen los problemas

Sentencias pegadas a partir de cadenas. Un cliente llamado O'Brien rompe una página de búsqueda, o una entrada maliciosa permite leer los datos de otra persona. La causa es insertar la entrada del usuario en el texto SQL, donde la base de datos no puede distinguir los datos de la orden. Envíe los valores como parámetros, separados de la sentencia, como permiten todos los controladores principales. En la propia biblioteca cliente de PostgreSQL, una llamada parametrizada también admite solo una orden por cadena, lo que añade una segunda línea de defensa contra la inyección.

Un orden que solo era casualidad. Una lista que llevaba meses llegando ordenada aparece mezclada tras una carga de datos o un plan nuevo. Sin ORDER BY, PostgreSQL devuelve las filas en el orden que resulte más rápido de producir, y un LIMIT sin ORDER BY puede incluso devolver un subconjunto distinto cada vez. Si el orden de un resultado importa, debe especificarse en la consulta.

Tratar NULL como un valor. Un filtro escrito como `WHERE discount = NULL` no devuelve nada, y `NOT IN` contra una subconsulta que contenga un NULL tampoco. Toda comparación con NULL produce un valor desconocido, en lugar de verdadero o falso, y una cláusula WHERE conserva solo las filas cuya condición es verdadera. Escriba `IS NULL` y prefiera `NOT EXISTS` a `NOT IN` cuando la columna interna pueda contener un NULL.

Un dialecto que se cuela en las pruebas. Una batería de pruebas pasa con SQLite y el mismo código falla con PostgreSQL en producción, o una migración a SQL Server tropieza con cada `LIMIT`, que ese motor expresa como `TOP` u `OFFSET ... FETCH`. Cada motor acepta cosas que los demás rechazan. Pruebe con el motor que vaya a desplegar y concentre la sintaxis específica de cada motor en el menor número de lugares posible.

Lecturas adicionales

- Parte II. El lenguaje SQL — El tutorial y la referencia completos de PostgreSQL para el lenguaje, desde la sintaxis hasta consejos de rendimiento.

- Apéndice D. Conformidad con SQL — Las partes de ISO/IEC 9075 y las características que implementa PostgreSQL.
- Referencia de Transact-SQL — El dialecto de SQL Server, documentado por Microsoft.
- Introducción a SQL en BigQuery — GoogleSQL y el dialecto anterior al que sustituyó.
- Lenguaje de consulta que entiende SQLite — La sintaxis de un motor integrado, sentencia a sentencia.

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 SQL?» 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/sql/