Engineer

PostgreSQL

base de datos de la mayor parte del trabajo presentado

Qué es

PostgreSQL es un servidor de bases de datos relacionales, gratuito y de código abierto, que guarda datos en tablas, responde a preguntas escritas en SQL y garantiza que cada transacción se confirme por completo o se revierta entera. La mayoría de quienes trabajan con él lo llaman Postgres, y el proyecto reconoce ambos nombres como propios. Ante la pregunta «¿qué es PostgreSQL?», un ingeniero en ejercicio lo diría sin rodeos: es la base de datos de propósito general con la que empieza un sistema nuevo cuando nadie tiene una razón concreta para elegir otra.

Es anterior a la mayoría del software que hoy funciona sobre él. El proyecto POSTGRES nació en la Universidad de California en Berkeley bajo la dirección del profesor Michael Stonebraker, y su implementación comenzó en 1986. Andrew Yu y Jolly Chen le incorporaron un intérprete de SQL en 1994 y publicaron el resultado como Postgres95. Dos años después pasó a llamarse PostgreSQL, y la numeración de versiones se retomó en 6.0 para continuar la secuencia iniciada en Berkeley. El propio proyecto describe el sistema como objeto-relacional: además de tablas y operaciones de combinación entre ellas, un usuario puede añadir tipos de datos, funciones, operadores, agregados, métodos de indexación y lenguajes procedimentales, y el servidor los trata como propios.

Su arquitectura consta de un cliente y un servidor. Un proceso del servidor controla los archivos de datos, y cada conexión de cliente recibe su propio proceso, creado mediante una bifurcación exclusivamente para esa conexión. Las aplicaciones le envían SQL por esa conexión mediante un controlador para el lenguaje en el que estén escritas. La concurrencia se apoya en el control de concurrencia multiversión: una actualización escribe una nueva versión de una fila en lugar de sobrescribir la anterior, de modo que cada sentencia lee una instantánea coherente, los lectores nunca esperan a los escritores y los escritores nunca esperan a los lectores.

¿Dónde se sitúa entre las alternativas? Junto a MySQL y MariaDB como los otros servidores relacionales de código abierto; junto a Oracle y Microsoft SQL Server como los comerciales; junto a SQLite como base de datos integrada que reside dentro de un programa; y frente a almacenes documentales como MongoDB, que guardan registros sin un esquema fijo.

Utilice el concepto en una decisión

Aplique «¿Qué es PostgreSQL?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Detectar las consultas y los índices que faltan en un sistema cada año más pesado y corregir lo que realmente importa.

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. 30 días de datos eléctricos en 180 ms — 180 ms para agregar un mes de lecturas de contadores

30 días de datos eléctricos en 180 ms Póngase en contacto →

Por qué importa

Elegir una base de datos es una decisión con la que un equipo convive durante años, porque los datos sobreviven al código escrito para utilizarlos. PostgreSQL es una opción predeterminada que da pocos motivos para arrepentirse, por tres razones que un comprador puede comprobar.

La primera es el coste y la dependencia del proveedor. Se publica bajo la PostgreSQL License, una licencia permisiva breve de la familia de BSD y MIT, por lo que no hay tarifas por núcleo, auditorías ni un proveedor que pueda cambiar las condiciones. Amazon, Microsoft y Google lo venden como servicio gestionado, y la misma base de datos funciona en un portátil, lo que significa que cambiar de host supone migrar los datos en lugar de reescribir la aplicación.

La segunda es la amplitud. Un servidor cubre tablas relacionales, documentos JSON en columnas `jsonb`, búsqueda de texto completo, consultas geoespaciales mediante la extensión PostGIS y tareas programadas mediante `pg_cron`. Cada una de esas funciones evita que el equipo tenga que operar, proteger y respaldar un producto independiente, además de contratar personal para él. Para la mayoría de los productos, la elección entre PostgreSQL y MySQL se decide aquí: para lecturas y escrituras sencillas ambos sirven bien, y la diferencia aparece cuando el trabajo necesita tipos más ricos, restricciones más estrictas o una extensión.

La tercera es la continuidad. El proyecto publica una versión principal al año y mantiene cada una durante cinco años, por lo que una actualización puede planificarse en lugar de venir impuesta por una emergencia. PostgreSQL 18, publicado en septiembre de 2025, añadió E/S asíncrona y una función nativa para UUID ordenados temporalmente, y el SQL escrito para versiones de una década antes suele ejecutarse sin cambios.

Las contrapartidas son reales. Un proceso por conexión hace que las conexiones sean costosas, por lo que una aplicación con mucha actividad necesita un gestor de pools de conexiones. Escala con más facilidad en una máquina más potente que entre varias: repartir las escrituras entre muchos servidores exige particionamiento, replicación o una extensión, y cada opción añade carga operativa. Además, sus valores predeterminados están ajustados para una máquina pequeña, de modo que una instalación de envergadura necesita una configuración deliberada.

Cómo se aplica en la práctica

Muchos de los proyectos de este sitio funcionan sobre PostgreSQL, y tres de ellos muestran tres formas distintas de apoyarse en él.

En una plataforma, la base de datos se convirtió en toda la capa de datos. Cada lectura y escritura pasa por una de las 1.275 funciones de PostgreSQL, repartidas entre 34 esquemas, y la API en Go no tiene acceso directo a ninguna tabla. El rol con el que la API inicia sesión, `mariner`, solo tiene EXECUTE sobre las funciones y USAGE sobre los esquemas, unos 4.000 permisos explícitos. Las funciones se ejecutan como SECURITY DEFINER bajo un propietario separado que no puede iniciar sesión y con un `search_path` fijado. El propio Postgres impone el límite de acceso; el precio es que un campo nuevo exige una migración y un cambio en una función.

En una aplicación que almacena organizaciones cliente, ubicaciones y lecturas diarias de electricidad, el trabajo consistió en ajustar el rendimiento, no en diseñar el sistema, y el encargo era multiplicarlo por 10 sin rehacer la arquitectura. Los índices GiST para los datos espaciales, los índices compuestos por organización cliente e intervalo de fechas, el particionamiento por rangos de la tabla de electricidad por día y mes y las consultas reescritas tras leer sus planes con `EXPLAIN ANALYZE` mejoraron los tiempos de respuesta de las consultas por un factor de 10 y redujeron el uso de recursos un 40 %.

Y en una plataforma con trabajo lento en segundo plano, la cola de trabajos reside en la misma base de datos. River guarda su cola en PostgreSQL, de modo que un trabajo y las filas que modifica se confirman o se revierten juntos. Detrás hay 15 módulos de procesamiento de trabajos, 8 tareas programadas y 20 trabajos de `pg_cron` que eliminan particiones antiguas, rotan las sales y actualizan agregados. La página expone el límite con claridad: a partir de cierto volumen una cola en Postgres deja de ser la respuesta adecuada, y tener un componente operativo menos valía más que un margen de capacidad que nadie iba a utilizar.

Dónde surgen los problemas

PostgreSQL rara vez pierde datos. Sí sorprende a quienes lo utilizan, y esas sorpresas suelen adoptar unas cuantas formas conocidas.

La tabla crece aunque el número de filas no cambie. Un `UPDATE` o `DELETE` deja atrás la versión anterior de la fila hasta que `VACUUM` la elimina, porque otra transacción podría seguir leyéndola. Una transacción que permanece abierta durante horas retrasa esa limpieza en toda la base de datos, y los valores predeterminados de autovacuum están pensados para tablas pequeñas. Cierre las transacciones inactivas, ajuste autovacuum por tabla en las grandes y particione los datos que caducan para poder descartar meses antiguos en lugar de borrar sus filas. Vacuum también cumple una segunda función: los identificadores de transacción tienen 32 bits, cada tabla debe pasar por vacuum al menos una vez cada dos mil millones de transacciones, y un servidor que se queda demasiado atrás rechaza nuevas escrituras para proteger los datos.

La memoria se agota en el primer pico de tráfico. Cada conexión es un proceso del servidor con su propia memoria, y el límite predeterminado es de 100 conexiones. Veinte instancias de aplicación con un pool de cincuenta conexiones cada una solicitan mil procesos. Coloque delante un gestor de pools de conexiones como PgBouncer y dimensione el pool de cada aplicación según el trabajo, que suele requerir solo unas pocas conexiones por instancia.

Una consulta que tardaba milisegundos empieza a tardar segundos. El planificador eligió otro plan a medida que crecían los datos, a menudo porque sus estadísticas quedaron obsoletas o los índices se habían diseñado para las consultas de un año antes. Lea el plan antes de cambiar nada, actualice las estadísticas con `ANALYZE` e indexe el predicado por el que la consulta filtra realmente:

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, total FROM orders
WHERE customer_id = 42 AND placed_at >= '2026-01-01';
```

Las inserciones se ralentizan a medida que crece la tabla. Los UUID aleatorios utilizados como claves primarias caen en puntos dispersos del índice B-tree, por lo que cada inserción toca una página distinta y el índice se fragmenta. Los identificadores ordenados temporalmente evitan esto: una plataforma convirtió UUID v7 en su clave predeterminada mediante `uuidv7()` de PostgreSQL 18, y las filas nuevas se insertan en orden dentro del índice en lugar de dispersarse por él.

Lecturas adicionales

- ¿Qué es PostgreSQL? — La definición del propio proyecto, con las funciones y los puntos de extensión que enumera.
- Breve historia de PostgreSQL — Berkeley, Postgres95 y el cambio de nombre, contados por quienes lo hicieron.
- Control de concurrencia: introducción — Cómo las instantáneas permiten que las lecturas y escrituras avancen sin bloquearse entre sí.
- Vacuum rutinario — Versiones muertas de filas, autovacuum y vuelta al inicio del contador de identificadores de transacción, en detalle.
- Uso de EXPLAIN — Cómo leer un plan de consulta, con ejemplos desarrollados.
- Política de versiones — La versión principal anual y los cinco años de soporte de cada una.

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 PostgreSQL?» 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/postgresql/