Engineer

Particionamiento

División de una tabla en partes físicas

Qué es

El particionamiento, también llamado particionamiento de tablas, almacena las filas de una tabla lógica en varias tablas ordinarias. Cada una contiene las filas cuya clave está dentro de sus límites, mientras las consultas y escrituras siguen usando el mismo nombre de siempre. PostgreSQL, en el que se basa esta página, llama particionamiento declarativo a su modalidad integrada: la tabla se declara particionada, con un método y una clave de partición, y cada partición se crea como una tabla propia.

La tabla particionada no contiene filas por sí misma. Una inserción se dirige a la partición cuyos límites corresponden a la clave, y una actualización que cambia la clave mueve la fila a la partición a la que ahora pertenece. Las particiones comparten las columnas de la tabla padre, pero pueden tener sus propios índices, ajustes de almacenamiento y espacio de tablas; una partición puede, a su vez, particionarse.

Hay tres métodos. El particionamiento por rangos asigna a cada partición un intervalo de valores, normalmente fechas; el límite inferior se incluye y el superior no, de modo que, con intervalos 1–10 y 10–20, el valor 10 pertenece al segundo. El particionamiento por listas enumera los valores que acepta cada partición, como un conjunto de códigos de país. El particionamiento por hash asigna a cada partición un módulo y un resto, y cada fila va a la partición cuyo resto coincide con el que deja el hash de su clave. Así se distribuyen las filas de manera uniforme cuando no hay un rango natural por el que dividirlas. Una partición predeterminada puede recoger los valores que no encajan en ninguna otra, solo en el particionamiento por rangos y por listas.

El particionamiento ocurre dentro de un único servidor de base de datos. Dividir los datos entre servidores separados, cada uno con su propia copia del esquema, es sharding, un paso mucho mayor. SQL Server y BigQuery tienen sus propias formas de particionamiento, con su propia sintaxis, y las ideas siguientes se trasladan a ellos aunque los comandos sean distintos.

Utilice el concepto en una decisión

Aplique «¿Qué es el particionamiento?» 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

La mayoría de las tablas grandes crecen por un extremo. Pedidos, lecturas, registros y eventos llegan cada día, permanecen meses o años, y la mayoría de las consultas piden la parte reciente. Sin particionamiento, todas las consultas trabajan sobre una tabla y un conjunto de índices que siguen creciendo.

Con una clave de partición que encaje con las consultas, el planificador puede demostrar que la mayoría de las particiones no pueden contener una fila coincidente y excluirlas por completo del plan. PostgreSQL lo llama poda de particiones. Un informe de la semana pasada lee la partición de esa semana y nada más. La documentación añade una ventaja menos evidente: como las particiones asumen la función de los niveles superiores del árbol de un índice, es más probable que las partes de los índices más utilizadas quepan en memoria.

La retirada de datos antiguos es lo que más cambia. Borrar de una sola tabla las filas de un mes exige un `DELETE` largo que deja filas muertas para que `VACUUM` las limpie. Con una partición por mes, la misma política de retención se aplica con un solo `DROP TABLE` o `DETACH PARTITION`. Según la documentación, esto es mucho más rápido que una operación masiva y evita por completo la sobrecarga de `VACUUM`. Una partición separada sigue siendo una tabla ordinaria, así que se puede hacer una copia de seguridad o un resumen antes de eliminarla. Las particiones que apenas se leen pueden alojarse en un almacenamiento más barato y lento.

La ventaja tiene un coste, y la documentación explica con claridad cuándo compensa. Cada partición añade trabajo de planificación, y las ventajas normalmente solo merecen la pena si, de otro modo, la tabla sería muy grande. Como regla orientativa, debería superar el tamaño de la memoria física del servidor de base de datos. El particionamiento tampoco cambia una cosa: una fila por la que compiten muchas transacciones sigue siendo una sola fila en una partición, bajo un único bloqueo.

Cómo se aplica en la práctica

Una tabla `orders` dividida por mes según `created_at` es el caso más sencillo, en una nota sobre cómo escalar una base de datos antes de distribuirla mediante sharding, que dedica una sección a particionar las tablas alrededor de una fila muy disputada. La clave primaria es `(id, created_at)` porque debe incluir la columna de partición. Una consulta que indica un rango de fechas solo lee las particiones incluidas en él, los índices del mes actual siguen siendo lo bastante pequeños para residir en memoria y la política de retención se aplica con un solo `DETACH PARTITION` en lugar de un `DELETE` de millones de filas. El límite es igual de claro: el particionamiento no acelera la fila disputada, y una consulta que no indica la clave lee todas las particiones.

Un proyecto de mejora del rendimiento de PostgreSQL para una aplicación con cuentas de usuario, organizaciones cliente, datos geográficos y consumo diario de electricidad dividió la tabla de electricidad en intervalos temporales por día y por mes, y utilizó particionamiento por hash para los datos geográficos. Se combinó con índices GiST y compuestos y con consultas reescritas; el resultado conjunto fue que los tiempos de respuesta de las consultas se dividieron por 10 y la utilización de recursos se redujo un 40 %.

A una escala mucho mayor, el particionamiento fue una técnica entre varias para al menos 100 bases de datos en PostgreSQL, SQL Server y BigQuery, con una gran cantidad de datos geográficos y de series temporales. El particionamiento de series temporales formaba parte del diseño del esquema en PostgreSQL y SQL Server, y BigQuery utilizaba tablas particionadas por tiempo y agrupadas mediante clustering. Junto con índices espaciales, vistas materializadas y caché, los tiempos de respuesta de las consultas para esos datos disminuyeron un 40–60 %.

Dónde surgen los problemas

El particionamiento traslada parte del trabajo del momento de la consulta al diseño, y la mayoría de los problemas proceden de decisiones de diseño tomadas sin tener en cuenta las consultas.

Las inserciones empiezan a fallar el primer día de un mes nuevo. Ninguna partición cubre el rango nuevo, y PostgreSQL rechaza una inserción en la tabla padre que no corresponde a ninguna partición existente. Cree las particiones con antelación desde un trabajo programado que genere el DDL, como sugiere la documentación, y configure una alerta cuando la partición más reciente se acerque a su final:

```
CREATE TABLE orders_2026_11
  PARTITION OF orders
  FOR VALUES FROM ('2026-11-01')
              TO ('2026-12-01');
```

Una partición predeterminada evita los errores, pero las filas de todos los meses sin cobertura acaban juntas en esa única tabla. Al asociar una partición nueva más adelante, se examina la partición predeterminada bajo un bloqueo `ACCESS EXCLUSIVE`, salvo que una restricción `CHECK` en esta ya excluya el rango nuevo. Trate una partición predeterminada no vacía como una alarma.

Las consultas son más lentas después de particionar. Las consultas habituales no filtran por la clave de partición, de modo que cada una lee todas las particiones y sus índices. Otra causa es el número de particiones: la documentación dice que el planificador funciona bien con hasta unos pocos miles, siempre que las consultas habituales descarten todas salvo unas pocas, y que el tiempo y la memoria de planificación aumentan más allá de esa cifra. Elija la clave a partir de las cláusulas `WHERE` que ya escribe la aplicación, compruebe con `EXPLAIN` que solo aparecen las particiones esperadas y mantenga un número proporcionado. Una partición por cliente es una trampa habitual, porque la cantidad crece con el negocio; la documentación sugiere en su lugar particionamiento por hash con un número razonable de particiones.

No se puede crear una clave primaria o una restricción de unicidad. En una tabla particionada, una restricción de unicidad o una clave primaria debe incluir todas las columnas de la clave de partición, y esta no puede contener

expresiones ni llamadas a funciones. El índice de cada partición solo puede imponer la unicidad dentro de ella. Añada la columna de la clave a la restricción, como hace la tabla `orders` de la nota con `(id, created_at)`.

Crear un índice en la tabla padre bloquea las escrituras durante mucho tiempo. Un `CREATE INDEX` normal en la tabla padre impide las escrituras hasta que se hayan indexado todas las particiones, y la opción `CONCURRENTLY` funciona en cada partición, no en la tabla padre. Cree el índice con `ON ONLY` en la tabla padre, lo que lo deja marcado como no válido, construya el índice de cada partición con `CONCURRENTLY` y asocie cada uno mediante `ALTER INDEX ... ATTACH PARTITION`. Cuando todos los índices de las particiones estén asociados, el índice padre pasa a ser válido por sí solo.

Una tabla grande existente no puede convertirse directamente. PostgreSQL no convierte una tabla ordinaria en una particionada. Cree la tabla particionada nueva y asocie la tabla antigua como una de sus particiones. Antes de asociarla, añada una restricción `CHECK` que coincida con los límites de la partición, para evitar que la asociación examine toda la tabla mientras mantiene en ella un bloqueo `ACCESS EXCLUSIVE`.

Lecturas adicionales

- Particionamiento de tablas — el capítulo de PostgreSQL: los tres métodos, la poda, el mantenimiento, las limitaciones y las buenas prácticas.
- CREATE TABLE — `PARTITION BY`, `PARTITION OF`, los límites de las particiones y la partición predeterminada.
- ALTER TABLE — `ATTACH PARTITION` y `DETACH PARTITION`, y los bloqueos de cada uno.
- CREATE INDEX — los índices de tablas particionadas, `ON ONLY` y la construcción con `CONCURRENTLY`.

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 el particionamiento?» 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/partitioning/