Engineer

Esquema

Espacio de nombres para objetos de una base de datos

Qué es

Un esquema es un espacio de nombres dentro de una base de datos: un contenedor de tablas, vistas, tipos de datos, funciones y operadores, que permite usar el mismo nombre en dos esquemas sin conflicto. El manual de PostgreSQL compara los esquemas con los directorios de un sistema de archivos, con la diferencia de que no se pueden anidar: una base de datos contiene esquemas y los esquemas contienen objetos. La palabra tiene un segundo sentido, menos estricto e igual de habitual: el esquema de una base de datos es toda su estructura, cada tabla, columna, clave y restricción. Ambos sentidos aparecen a continuación, y el plural, esquemas, casi siempre se refiere a los espacios de nombres.

La parte de facturación de una aplicación puede residir en su propio esquema, y cada sesión puede elegir en qué esquemas buscar:

```
CREATE SCHEMA billing;

CREATE TABLE billing.invoices (
  id     bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  total  numeric(12, 2) NOT NULL
);

SHOW search_path;                 -- "$user", public
SET search_path = billing, public;
SELECT count(*) FROM invoices;     -- now resolves to billing.invoices
```

Un nombre que incluye su esquema, como `billing.invoices`, siempre se refiere a la misma tabla. Un nombre sin él se busca a través de search_path, una lista de esquemas que el servidor recorre en orden hasta encontrar una coincidencia. La lista predeterminada es `"$user", public`: primero un esquema con el nombre del

rol actual, si existe, y después el esquema public con el que comienza cada nueva base de datos. El primer esquema existente de la ruta también es donde CREATE TABLE coloca una tabla cuyo nombre no incluye esquema. Por eso, los objetos terminan en public si nadie indica otra cosa. Un nombre que no se encuentra en ninguno de los esquemas enumerados provoca un error, aunque otro esquema de la misma base de datos lo contenga.

Utilice el concepto en una decisión

Aplique «¿Qué es un esquema?» 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

El primer motivo es la organización. Una base de datos con unos cientos de tablas necesita un nivel por encima de ellas, y los esquemas lo proporcionan: facturación en uno, cuentas en otro y las tablas de una herramienta de terceros en un tercero, sin conflictos con los nombres de los demás. Una conexión de cliente llega a una sola base de datos, pero los esquemas no están separados de forma rígida: si los privilegios lo permiten, un rol puede usar objetos de cualquier esquema de la base de datos a la que está conectado, y una consulta puede unir tablas de varios.

El segundo motivo es el control de acceso. USAGE sobre un esquema permite a un rol buscar los objetos que contiene, y CREATE permite añadir otros nuevos. Retirar USAGE bloquea el acceso a los objetos, aunque sus nombres siguen siendo legibles en los catálogos del sistema. Por tanto, un esquema protege los datos, no el conocimiento de lo que existe. `CREATE SCHEMA name AUTHORIZATION role` asigna la propiedad de un esquema nuevo a un rol, y los nombres que empiezan por `pg_` están reservados al sistema.

El tercer motivo es la seguridad, porque los esquemas intervienen en la resolución de nombres. La mayoría de los nombres de las consultas se escribe sin esquema. Por eso, en palabras del manual, añadir un esquema a search_path equivale a confiar en todos los roles que puedan crear objetos en él. Uno de esos roles puede hacer que una consulta normal ejecute funciones elegidas por él,

como si las hubiera llamado la persona que ejecuta la consulta. Los patrones seguros excluyen de la ruta de un usuario los esquemas en los que otros pueden escribir. Un ejemplo es un esquema privado por usuario, creado con `CREATE SCHEMA alice AUTHORIZATION alice`, que la entrada `"$user"` encuentra primero. PostgreSQL 15 orientó las nuevas bases de datos en esa dirección: retiró el derecho de todos los roles a crear objetos en public e hizo que el propietario de la base de datos, a través del rol pg_database_owner, fuera también propietario de public.

La portabilidad es el último punto. El estándar SQL tiene esquemas, pero no un esquema public, y el manual señala que algunos sistemas SQL podrían no implementar esquemas en absoluto. Los nombres calificados se adaptan mejor a otros sistemas que el código que depende de search_path.

Cómo se aplica en la práctica

El modelo de datos de 348 tablas modeladas en 34 esquemas de Postgres utiliza los esquemas en el primer sentido, como nivel por encima de las tablas. El dominio marítimo, con sus profesionales, empresas, barcos, empleos y reseñas, se modela mediante esquemas normalizados de PostgreSQL organizados por dominio: 348 tablas repartidas entre 34 esquemas. Los nombres de las tablas están en plural dentro de esquemas con nombres en singular, sin excepción. Esto también evita muchas colisiones con palabras reservadas. Las relaciones se sostienen con claves foráneas y restricciones reales. Añadir una funcionalidad suele significar ampliar el esquema siguiendo su estructura existente en lugar de luchar contra ella, algo que la página presenta como la única razón por la que 34 esquemas forman una estructura y no una maraña.

Un esquema también puede delimitar dos responsabilidades. En la implementación de derechos de acceso para Stripe y las compras dentro de la aplicación, los pagos con tarjeta de Stripe y las compras dentro de la aplicación gestionadas por Apple y Google convergen en un esquema de pagos con 9 tablas y 34 funciones. Un esquema de acceso independiente, con 11 tablas y 34 funciones, responde «¿puede esta cuenta hacer esto?» sin saber en qué tienda se pagó. Esa respuesta, comprobada de nuevo en el servidor en cada petición, controla el acceso al directorio de empresas, la búsqueda y la exportación de datos. El coste, como explica la página, son dos esquemas donde un producto más pequeño querría uno, además de una consulta de derechos de acceso en peticiones que, de otro modo, no tendrían ese coste.

El segundo sentido de la palabra es el tema de 1.022 migraciones protegidas por un control de divergencia del esquema. La mayoría de los proyectos describe el esquema dos veces: en la configuración inicial que lo construye desde cero y en las migraciones que lo hicieron crecer. Ambas divergen si nada las comprueba. Aquí la base de datos se versiona mediante 1.022 migraciones, numeradas de 036 a 1102. Un trabajo de CI la construye dos veces en cada cambio, una desde cero y otra reproduciendo todas las migraciones en orden, y después compara los dos resultados: tanto los datos iniciales como la estructura. Una migración que completa incorrectamente los datos de una tabla de referencia hace tanto daño como otra que olvida una columna, pero solo una aparece en un diff del esquema. Cualquier discrepancia hace fallar la compilación e imprime la diferencia. Por eso, cada migración debe funcionar tanto al reproducirse como al partir de cero.

Dónde surgen los problemas

Una función infiltrada en un esquema compartido. La guía de la wiki de PostgreSQL sobre CVE-2018-1058 muestra el ataque: un usuario crea `lower(varchar)` en public. La función incorporada `lower` está en pg_catalog y acepta text, de modo que, para una columna varchar, la función infiltrada es la coincidencia más cercana. Un compañero que ejecuta `SELECT lower(full_name) FROM a` ejecuta el código infiltrado. Califique las llamadas en código sensible como `pg_catalog.lower` , retire CREATE sobre public a los roles que no lo necesiten y enumere lo que ya existe con `\df public.*` .

Una actualización conserva los permisos antiguos. La configuración predeterminada más segura de PostgreSQL 15 se aplica a clústeres y bases de datos de nueva creación. Actualizar un clúster o restaurar un volcado conserva los permisos que ya tenía public, por lo que un servidor con la versión 15 o posterior todavía puede permitir que todos los roles creen objetos allí. El manual considera que la ruta predeterminada con CREATE concedido sobre public nunca es un patrón seguro, y señala que es la predeterminada en bases de datos actualizadas desde la versión 14 o anterior. En una base de datos así, ejecute `REVOKE CREATE ON SCHEMA public FROM PUBLIC` y después audite public en busca de objetos con nombres como los de pg_catalog.

Una tabla se crea en el lugar equivocado. Una migración crea `invoices` sin esquema y la tabla termina en el primer esquema existente de la ruta del rol que la ejecuta. La aplicación, conectada con otra ruta, lee entonces otra tabla o recibe `relation "invoices" does not exist` . Califique todos los nombres de las migraciones y compruebe `SELECT current_schema()` en la sesión que las ejecuta.

Un borrado llega más allá del esquema. `DROP SCHEMA billing` se niega a continuar mientras el esquema contenga algo, porque RESTRICT es la opción predeterminada. Con CASCADE elimina todos los objetos de dentro y los que dependen de ellos. El manual advierte que esto puede eliminar objetos de otros esquemas distintos del indicado. Aconseja ejecutar primero DROP sin CASCADE y leer las líneas DETAIL, que enumeran lo que se eliminaría.

Un nombre que el sistema resuelve primero. Si search_path no incluye pg_catalog, este se busca implícitamente antes que los esquemas enumerados en la ruta. Si se incluye de forma explícita, su posición determina el orden de búsqueda. Una tabla propia cuyo nombre empieza por `pg_` puede quedar oculta si una versión posterior añade una tabla del sistema con el mismo nombre, por lo que el manual desaconseja esos nombres. Las palabras reservadas como `user` y `order` causan una versión más leve del problema: una tabla con uno de esos nombres necesita comillas dobles en todas las sentencias.

Lecturas adicionales

- 5.10. Esquemas — El capítulo del manual: nombres calificados, esquema public, search_path, patrones de uso seguro y portabilidad.
- CREATE SCHEMA — Sintaxis del comando, cláusula AUTHORIZATION, prefijo reservado `pg_` y diferencias de PostgreSQL respecto al estándar.
- 5.8. Privilegios — Qué significan USAGE y CREATE sobre un esquema, junto con los demás privilegios.
- E.20. Versión 15 — La versión que retiró CREATE sobre public en las nuevas bases de datos y lo conservó en las actualizadas.
- Guía sobre CVE-2018-1058: proteja su ruta de búsqueda — La explicación del proyecto sobre el ataque mediante una función infiltrada, con las correcciones y los comandos para auditar una base de datos.

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 un esquema?» 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/schema/