Engineer

Схема бази даних

названа група таблиць в одній базі даних

Що це таке

Схема бази даних — це простір імен усередині бази даних: контейнер для таблиць, представлень, типів даних, функцій і операторів, завдяки якому те саме ім’я можна використати у двох схемах без конфлікту. Посібник PostgreSQL порівнює схеми з каталогами файлової системи, з тією різницею, що вкладати їх одна в одну не можна: база даних містить схеми, а схеми містять об’єкти. Слово має й друге, ширше значення, не менш поширене: схема бази даних — це вся її структура, кожна таблиця, колонка, ключ і обмеження, і зміна в схемі означає зміну будь-чого з цього. Нижче трапляються обидва значення. Коли текст описує схему як структуру, це видно з контексту, а «кілька схем» майже завжди означає простори імен.

Білінг застосунку може жити у власній схемі, а сесія може вибрати, у яких схемах шукати:

```
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;     -- тепер це billing.invoices
```

Ім’я, записане разом зі схемою, як-от `billing.invoices`, завжди означає ту саму таблицю. Ім’я без схеми шукається через search_path — список схем, які сервер обходить по черзі, доки не знайде збіг. Типовий список — `"$user", public`: спершу схема з іменем поточної ролі, якщо така існує, потім схема

public, з якою починається кожна нова база даних. Перша наявна схема зі шляху — це також місце, куди CREATE TABLE кладе таблицю, ім’я якої записане без схеми, тому об’єкти потрапляють у public, якщо хтось не вкаже інакше. Якщо ім’я є лише в схемі поза списком, пошук завершується помилкою, навіть коли таблиця існує в тій самій базі даних.

Застосуйте поняття до рішення

Застосуйте «Що таке схема бази даних?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Структура, що відповідає реальній роботі бізнесу, тож потрібні згодом запитання до даних усе ще можна буде поставити.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Спроєктувала й підтримувала 6 пайплайнів ETL/ELT — 6 пайплайнів ETL на трьох рушіях баз даних

Спроєктувала й підтримувала 6 пайплайнів ETL/ELT Зв'язатися з нами →

Чому це важливо

Перша причина — порядок. Базі даних із кількома сотнями таблиць потрібен рівень над таблицею, і схеми його дають: білінг в одній, облікові записи в іншій, таблиці стороннього інструмента в третій. Клієнтське з’єднання дістається лише однієї бази даних, але схеми не відокремлені жорстко: роль може користуватися об’єктами в будь-якій схемі бази, до якої під’єднана, якщо дозволяють права, а один запит може з’єднувати таблиці з кількох схем.

Друга причина — контроль доступу. Право USAGE на схему дозволяє ролі знаходити об’єкти в ній, а CREATE — створювати нові. Якщо забрати USAGE, доступ до об’єктів закривається, але їхні імена й далі можна прочитати в системних каталогах, тож схема захищає дані, а не знання про те, що існує. `CREATE SCHEMA name AUTHORIZATION role` передає нову схему ролі, а імена, що починаються з `pg_`, зарезервовані для системи.

Третя причина — безпека, бо схеми беруть участь у пошуку імен. Більшість імен у запитах пишуть без схеми, тож, за словами посібника, додати схему до search_path фактично означає довіряти кожній ролі, яка може створювати в ній об’єкти. Така роль може змусити звичайний запит виконати функції на її вибір так, ніби їх викликав той, хто виконує запит. Безпечні шаблони тримають схеми, у які можуть писати інші, поза шляхом користувача, наприклад по одній приватній схемі на користувача, створеній командою `CREATE SCHEMA alice AUTHORIZATION alice`, яку елемент `"$user"` тоді знаходить

першою. PostgreSQL 15 зрушив нові бази даних у цей бік: версія забрала в усіх ролей право створювати об’єкти в public і зробила власника бази даних, через роль pg_database_owner, власником public.

Остання причина — переносимість. Стандарт SQL має схеми, але не має схеми public, а посібник зауважує, що деякі системи SQL можуть узагалі не реалізовувати схем. Кваліфіковані імена переносяться краще, ніж код, що спирається на search_path.

Як це виглядає на практиці

Модель даних у Змоделювала морський домен у 348 таблицях і 34 схемах використовує схеми в першому значенні, як рівень над таблицею. Морський домен із фахівцями, компаніями, суднами, вакансіями й відгуками організовано як нормалізовані схеми PostgreSQL, згруповані за доменами, — 348 таблиць у 34 схемах. Таблиці мають імена в множині всередині схем з іменами в однині, без винятків, що також дає змогу уникнути багатьох конфліктів із зарезервованими словами, а зв’язки підтримують реальні зовнішні ключі та обмеження. Додавання функції зазвичай означає розширення схеми за наявною логікою, а не боротьбу проти неї, і сторінка називає це єдиною причиною, чому 34 схеми — це структура, а не розповзання.

Схема може бути й межею між двома завданнями. У Побудувала рівень платежів і прав доступу карткові платежі Stripe та in-app-покупки Apple і Google сходяться в схемі payments із 9 таблиць і 34 функцій. Окрема схема access, 11 таблиць і 34 функції, відповідає на «чи можна цьому обліковому запису зробити це?», не знаючи, через який магазин надійшла оплата, і ця відповідь, яку сервер перевіряє повторно на кожен запит, регулює доступ до каталогу компаній, пошуку й експорту даних. Ціна, як пише сторінка, — дві схеми там, де меншому продуктові вистачило б однієї, плюс перевірка прав на запитах, які інакше були б безкоштовними.

Друге значення слова — тема сторінки Тримала схему узгодженою впродовж 1 022 міграцій. У більшості проєктів схема описана двічі: у початковому налаштуванні, яке будує її з нуля, і в міграціях, які її виростили, а без перевірки ці описи розходяться. Тут база даних версіонована як 1 022 міграції, пронумеровані від 036 до 1102, і CI-задача на кожну зміну збирає її двічі, один раз зі свіжої схеми, а другий з кожною міграцією, відтвореною по черзі, і порівнює обидві: засіяні дані так само, як структуру. Міграція, яка

неправильно заповнює довідкову таблицю, шкодить так само, як та, що забула колонку, а в diff схеми потрапляє тільки одна з них. Будь-яка розбіжність валить білд із виведеною різницею, тож кожна міграція має працювати у відтворенні й з холодного старту.

Де виникають проблеми

Функція, підкинута в спільну схему. Розбір CVE-2018-1058 у вікі PostgreSQL показує атаку: один користувач створює `lower(varchar)` у public. Вбудована `lower` міститься в pg_catalog і приймає text, тож для колонки varchar підкинута функція підходить точніше, і колега, який виконує `SELECT lower(full_name) FROM a`, запускає підкинутий код. Кваліфікуйте виклики в чутливому коді як `pg_catalog.lower`, заберіть CREATE на public у ролей, яким воно не потрібне, і перегляньте, що там уже лежить, командою `\df public.*`.

Оновлення, що зберегло старі права. Безпечніша типова поведінка PostgreSQL 15 стосується нових кластерів і щойно створених баз даних. Оновлення кластера або відновлення дампа зберігає права, які public уже мала, тож сервер версії 15 чи новішої все ще може дозволяти кожній ролі створювати там об’єкти. Посібник називає типовий шлях разом із правом CREATE на public шаблоном, який ніколи не буває безпечним, і зауважує, що саме він типовий у базах даних, оновлених із версії 14 або давнішої. У такій базі виконайте `REVOKE CREATE ON SCHEMA public FROM PUBLIC`, а потім перевірте public на об’єкти з іменами, схожими на імена в pg_catalog.

Таблиця, створена не там. Міграція створює `invoices` без схеми, і таблиця потрапляє в першу наявну схему на шляху тієї ролі, яка запускає міграцію. Застосунок, що під’єднується з іншим шляхом, тоді читає іншу таблицю або отримує `relation "invoices" does not exist`. Кваліфікуйте кожне ім’я в міграціях і перевіряйте `SELECT current_schema()` у сесії, яка їх запускає.

DROP, що сягає за межі схеми. `DROP SCHEMA billing` відмовляє, поки в схемі щось є, бо типовим є RESTRICT. З CASCADE команда видаляє всі об’єкти в схемі й усі об’єкти, що від них залежать, і посібник попереджає, що так можна видалити об’єкти й в інших схемах, крім названої. Порада посібника — спершу виконати DROP без CASCADE і прочитати рядки DETAIL, які перелічують, що буде видалено.

Ім’я, яке система знаходить першим. Якщо pg_catalog не зазначено в search_path, пошук у ньому неявно відбувається раніше, ніж у схемах, перелічених на шляху. Якщо його зазначено явно, порядок пошуку визначає його позиція на шляху. Вашу власну таблицю з іменем, що починається з `pg_`, може затінити системна таблиця з тим самим іменем, яку додасть пізніша версія, тому посібник радить уникати таких імен. Зарезервовані слова, як-от `user` і `order`, дають м’якшу версію тієї самої проблеми: таблиця з таким іменем потребує подвійних лапок у кожній інструкції.

Що почитати далі

- 5.10. Schemas — розділ посібника: кваліфіковані імена, схема public, search_path, безпечні шаблони використання й переносимість.
- CREATE SCHEMA — синтаксис команди, конструкція AUTHORIZATION, зарезервований префікс `pg_` і місця, де PostgreSQL відходить від стандарту.
- 5.8. Privileges — що означають USAGE і CREATE для схеми, поряд з усіма іншими правами.
- E.20. Release 15 — випуск, який забрав CREATE на public у нових базах даних і зберіг його в оновлених.
- A Guide to CVE-2018-1058: Protect Your Search Path — розбір атаки з підкинутою функцією від самого проєкту, з виправленнями й командами для перевірки бази даних.

Словник

Обирайте за фактичними умовами

Надайте репрезентативний набір даних, запитання до нього та поточне вузьке місце. Узгодимо вимірювання якості, пропускної здатності та відновлення.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке схема бази даних?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/schema/