Engineer

Databaseskema

en navngiven gruppe tabeller i én database

Hvad det er

Et databaseskema er et navnerum inde i en database: en beholder til tabeller, views, datatyper, funktioner og operatorer, så det samme navn kan bruges i to skemaer uden konflikt. PostgreSQL's manual sammenligner skemaer med mapper i et filsystem, bortset fra at de ikke kan ligge inden i hinanden: En database rummer skemaer, og skemaer rummer objekter. Ordet har også en anden, lige så almindelig betydning, hvor skemaet er hele databasens struktur, hver tabel, kolonne, nøgle og constraint, og skemaets historie er summen af de ændringer, der har formet det. Begge betydninger bruges nedenfor, og skemaer i flertal betyder næsten altid navnerummene.

Et faktureringsområde i en applikation kan bo i sit eget skema, og en session kan vælge, hvilke skemaer den kigger i:

```
CREATE SCHEMA fakturering;

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

SHOW search_path;                       -- "$user", public
SET search_path = fakturering, public;
SELECT count(*) FROM fakturaer;         -- nu fakturering.fakturaer
```

Et navn skrevet med sit skema, som `fakturering.fakturaer`, betyder altid den samme tabel. Et navn uden skema slås op gennem search_path, en liste af skemaer, som serveren går igennem i rækkefølge, til den finder et match. Standardlisten er `"$user", public`: et skema med samme navn som den aktuelle rolle, hvis det findes, og derefter det public-skema, som enhver ny database

starter med. Det første eksisterende skema på stien er også dér, hvor CREATE TABLE lægger en tabel, hvis navn står uden skema, og derfor havner objekter i public, medmindre nogen siger andet. Står navnet kun i et skema uden for listen, giver opslaget en fejl, også selv om tabellen findes i samme database.

Brug begrebet i en beslutning

Brug »Hvad er et databaseskema?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. En struktur, der passer til, hvordan forretningen faktisk fungerer, så de svar, I får brug for senere, stadig kan hentes.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

Den første grund er orden. En database med et par hundrede tabeller har brug for et niveau over tabellen, og skemaer giver det: fakturering i ét, konti i et andet, et tredjepartsværktøjs tabeller i et tredje. En klientforbindelse når kun én database, men skemaer er ikke skarpt adskilt: En rolle kan bruge objekter i ethvert skema i den database, den er forbundet til, hvis rettighederne tillader det, og én SQL-forespørgsel kan joine tabeller fra flere af dem.

Den anden grund er adgangskontrol. USAGE på et skema lader en rolle slå objekterne i det op, og CREATE lader den oprette nye. Fjerner man USAGE, spærres adgangen til objekterne, men deres navne kan stadig læses i systemkatalogerne, så et skema beskytter data og ikke viden om, hvad der findes. `CREATE SCHEMA navn AUTHORIZATION rolle` giver et nyt skema til en rolle, og navne, der begynder med `pg_`, er reserveret til systemet.

Den tredje grund er sikkerhed, fordi skemaer indgår i navneopslag. De fleste navne i forespørgsler skrives uden skema, så det at føje et skema til search_path svarer ifølge manualen reelt til at stole på enhver rolle, der kan oprette objekter i det. En sådan rolle kan få en almindelig forespørgsel til at køre funktioner efter eget valg, som om den, der kørte forespørgslen, selv havde kaldt dem. De sikre mønstre holder skemaer, som andre kan skrive i, væk fra en brugers sti, for eksempel med ét privat skema pr. bruger, oprettet med `CREATE SCHEMA alice AUTHORIZATION alice`, som `"$user"`-posten så finder først. PostgreSQL 15 flyttede

nye databaser i den retning: Versionen fjernede alle rollers ret til at oprette objekter i public og gjorde databasens ejer, gennem rollen pg_database_owner, til ejer af public.

Portabilitet er det sidste punkt. SQL-standarden har skemaer, men intet public-skema, og manualen bemærker, at nogle SQL-systemer måske slet ikke implementerer skemaer. Kvalificerede navne rejser bedre end kode, der læner sig op ad search_path.

Sådan ser det ud i praksis

Datamodellen i Har modelleret 348 tabeller i 34 Postgres-skemaer bruger skemaer i den første betydning, som niveauet over tabellen. Det maritime domæne med professionelle, virksomheder, skibe, jobs og anmeldelser er lagt ud som normaliserede PostgreSQL-skemaer organiseret efter domæne: 348 tabeller på tværs af 34 af dem. Tabelnavne står i flertal inde i entalsnavngivne skemaer, uden undtagelse, hvilket også undgår en masse kollisioner med reserverede ord, og rigtige foreign keys og constraints holder relationerne oppe. At tilføje en funktion betyder som regel at udvide skemaet med den eksisterende retning frem for imod den, og det kalder siden den eneste grund til, at 34 skemaer er en struktur frem for et vildnis.

Et skema kan også være grænsen mellem to opgaver. I Har bygget Stripe- og in-app purchase-rettigheder løber Stripe-kortbetalinger og Apple- og Google-in-app-køb alle sammen i et payments-skema på 9 tabeller og 34 funktioner. Et separat access-skema på 11 tabeller og 34 funktioner svarer på »må denne konto det her?« uden at vide, hvilken butik der har betalt, og det svar, der genkontrolleres på serveren ved hver request, styrer adgangen til virksomhedskataloget, søgningen og dataeksporten. Prisen er ifølge siden to skemaer, hvor et mindre produkt ville nøjes med ét, plus et rettighedsopslag på requests, der ellers ville have været gratis.

Ordets anden betydning er emnet for Har holdt 1.022 migrationer fri for skemadrift. De fleste projekter beskriver et skema to gange, i den indledende opsætning, der bygger det fra bunden, og i de migrationer, der har fået det til at vokse, og de to driver fra hinanden, medmindre noget tjekker dem. Her er skemaets historie 1.022 migrationer lang, fra nummer 036 til 1102, og et CI-job bygger databasen to gange ved hver ændring, én gang frisk og én gang med hver eneste migration afspillet i rækkefølge, og sammenligner så de to: de seedede data såvel som strukturen. En migration, der backfiller en opslagstabel forkert, er

lige så skadelig som en, der glemmer en kolonne, og kun den ene af dem dukker op i en skema-diff. Afviger de to skemaer, fejler buildet med forskellen skrevet ud, så hver migration skal virke afspillet og fra koldt.

Hvor det går galt

En funktion plantet i et delt skema. PostgreSQL-wikiens guide til CVE-2018-1058 viser angrebet: Én bruger opretter `lower(varchar)` i public. Den indbyggede `lower` ligger i pg_catalog og tager text, så for en varchar-kolonne er den plantede funktion det tætteste match, og en kollega, der kører `SELECT lower(full_name) FROM a`, kører den plantede kode i stedet. Kvalificér kald i følsom kode som `pg_catalog.lower`, fjern CREATE på public fra de roller, der ikke har brug for det, og se, hvad der allerede ligger der, med `\df public.*`.

En opgradering, der beholdt de gamle rettigheder. PostgreSQL 15's sikrere standard gælder nye clusters og nyoprettede databaser. En opgradering af et cluster eller en gendannelse af et dump bevarer de rettigheder, public allerede havde, så en server på version 15 eller nyere stadig kan lade alle roller oprette objekter dér. Manualen kalder standardstien med CREATE tildelt på public et mønster, der aldrig er sikkert, og bemærker, at det er standarden i databaser opgraderet fra 14 eller tidligere. Kør i sådan en database `REVOKE CREATE ON SCHEMA public FROM PUBLIC`, og gennemgå derefter public for objekter med navne, der ligner dem i pg_catalog.

En tabel oprettet det forkerte sted. En migration opretter `fakturaer` uden skema, og tabellen havner i det første eksisterende skema på stien for den rolle, der kører den. Applikationen, der forbinder med en anden sti, læser så en anden tabel eller får `relation "fakturaer" does not exist`. Kvalificér hvert navn i migrationer, og tjek `SELECT current_schema()` i den session, der kører dem.

En DROP, der rækker ud over skemaet. `DROP SCHEMA fakturering` afviser, så længe skemaet indeholder noget, fordi RESTRICT er standarden. Med CASCADE fjerner den alle objekter i skemaet og alle objekter, der afhænger af dem, og manualen advarer om, at det kan fjerne objekter i andre skemaer end det navngivne. Rådet er at køre DROP uden CASCADE først og læse DETAIL-linjerne, der viser, hvad der ville forsvinde.

Et navn, systemet slår op først. Når pg_catalog er udeladt af search_path, gennemsøges det implicit før de skemaer, stien angiver. Er det nævnt eksplicit, afgør dets placering på stien søgerækkefølgen. En af dine egne tabeller, hvis

navn begynder med `pg_`, kan blive skygget, når en senere version tilføjer en systemtabel med samme navn, så manualen fraråder den slags navne. Reserverede ord som `user` og `order` giver en mildere udgave af problemet: En tabel med sådan et navn kræver dobbelte anførselstegn i hver sætning.

Læs mere

- 5.10. Schemas — manualens kapitel: kvalificerede navne, public-skemaet, search_path, sikre brugsmønstre og portabilitet.
- CREATE SCHEMA — kommandoens syntaks, AUTHORIZATION-klausulen, det reserverede `pg_` -præfiks og hvor PostgreSQL afviger fra standarden.
- 5.8. Privileges — hvad USAGE og CREATE betyder på et skema, side om side med alle andre rettigheder.
- E.20. Release 15 — udgivelsen, der fjernede CREATE på public fra nye databaser og beholdt det for opgraderede.
- A Guide to CVE-2018-1058: Protect Your Search Path — projektets gennemgang af angrebet med den plantede funktion, med rettelserne og kommandoerne til at gennemgå en database.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er et databaseskema?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/schema/