Engineer

スキーマ

一つのデータベース内で、テーブルなどを名前付きでまとめる領域

どのようなものか

スキーマは、データベース内の名前空間です。テーブル、ビュー、データ型、関数、演算子を入れる容器であり、二つのスキーマで同じ名前を使っても衝突しません。PostgreSQLのマニュアルは、ファイルシステムのディレクトリーに例えていますが、入れ子にできない点が異なります。データベースがスキーマを持ち、スキーマがオブジェクトを持ちます。同じくらいよく使う、もう一つの広い意味もあります。データベースのスキーマとは、テーブル、列、キー、制約をすべて含む構造全体のことです。以下では両方の意味を扱いますが、複数形のschemasは、ほぼ常に名前空間を指します。

アプリケーションの請求処理を専用のスキーマに置き、セッションで、どのスキーマから探すかを選べます。

```
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
```

`billing.invoices` のように、スキーマを付けた名前は、常に同じテーブルを指します。付けない名前は、search_pathを通じて探します。サーバーが一致するものを見つけるまで、順にたどるスキーマの一覧です。既定の一覧は `"$user", public` で、現在のロールと同名のスキーマが存在すればそこを探し、次に、新しいデータベースが最初から持つpublicスキーマを探します。CREATE TABLEで、名前にスキーマを付けないテーブルを作る場所も、経路上で最初に存在するスキ

ーマです。そのため、別の指定がなければオブジェクトはpublicに入ります。一覧のどこにも名前がなければ、同じデータベースの別のスキーマに存在していても、エラーになります。

概念を判断に生かす

「スキーマとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 実際の業務に合った構造を設計し、将来必要になる問いにも答えられるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 ETL/ELTパイプライン6本を設計・運用 — 6 3種類のDBエンジンにまたがるETLパイプライン

ETL/ELTパイプライン6本を設計・運用 お問い合わせ →

なぜ重要なのか

一つ目の理由は整理です。数百のテーブルを持つデータベースには、その上の階層が必要で、スキーマがその役割を担います。請求は一つ、アカウントは別、外部ツールのテーブルはさらに別のスキーマに分け、互いの名前が衝突しないようにします。クライアントの接続先は一つのデータベースですが、スキーマ同士は厳密に分断されていません。権限があれば、接続先のデータベース内のどのスキーマのオブジェクトも使え、一つのクエリーで複数のスキーマのテーブルを結合できます。

二つ目はアクセス制御です。スキーマのUSAGE権限は、中のオブジェクトを名前で探せるようにし、CREATE権限は新しいオブジェクトを追加できるようにします。USAGEを外すとオブジェクトへのアクセスを止められますが、名前はシステムカタログから引き続き読めます。つまり、スキーマが守るのはデータであり、何が存在するかという情報ではありません。`CREATE SCHEMA name AUTHORIZATION role` は、新しいスキーマをそのロールに所有させます。`pg_` で始まる名前は、システム用に予約されています。

三つ目は、スキーマが名前解決に関わることによるセキュリティーです。クエリー中の名前の多くは、スキーマを付けずに書きます。そのため、マニュアルによれば、スキーマをsearch_pathに加えることは、そこにオブジェクトを作れるすべてのロールを、実質的に信頼することになります。そのようなロールは、通常のクエリーに、自分の選んだ関数を、クエリーの実行者が呼んだかのように実行させられます。安全な使い方は、他人が書き込めるスキーマを利用者の検索経路から外すことです。たとえば、`CREATE SCHEMA alice AUTHORIZATION alice` で、利用者ごとに非公開の専用スキーマを作れば、`"$user"` がそれを最初に見つけます。PostgreSQL 15は、新しいデータベースの既定値を、その方向に変えました。す

べてのロールがpublicにオブジェクトを作れる権利をなくし、pg_database_ownerロールを介して、データベースの所有者をpublicの所有者にしました。

最後は移植性です。SQL標準にはスキーマがありますが、publicスキーマはありません。マニュアルは、スキーマをまったく実装しないSQLシステムもあり得るとしています。スキーマを付けた名前は、search_pathに頼るコードより移植しやすくなります。

実際の活用例

348のテーブルを34のPostgresスキーマにモデル化した実績のデータモデルは、テーブルの上の階層という、最初の意味でスキーマを使います。専門職、企業、船舶、求人、レビューを持つ海事の業務領域を、分野ごとに整理した、正規化されたPostgreSQLスキーマとして構成しています。348のテーブルを34のスキーマに配置しています。例外なく、単数形の名前を持つスキーマ内で、テーブルには複数形の名前を使い、予約語との衝突も多く避けています。また、実際の外部キーと制約で関係を保ちます。機能を追加するときも、通常は構造に逆らわず、既存の流れに沿ってスキーマを拡張できます。同実績は、これこそが、34のスキーマを乱立させず、一つの構造として保てる唯一の理由だと述べています。

スキーマは、二つの仕事の境界にもなります。Stripeとアプリ内購入の利用権限を構築した実績では、Stripeのカード決済、AppleとGoogleのアプリ内購入がすべて、9テーブル、34関数のpaymentsスキーマに集まります。別のaccessスキーマは11テーブル、34関数を持ち、どの販売窓口で支払ったかを知らずに、「このアカウントは、この操作をしてよいか」に答えます。その答えをサーバーがリクエストごとに再確認し、企業一覧、検索、データのエクスポートの利用を制御します。同実績が述べる代償は、より小さな製品なら一つで済ませたいところに二つのスキーマがあることと、そうでなければ追加の処理が不要だったリクエストにも、利用権限の照会が加わることです。

言葉の二つ目の意味を扱うのが、スキーマのドリフト検査で守る1,022のマイグレーションです。ほとんどのプロジェクトでは、ゼロから作る初期構築と、成長させてきたマイグレーションの二通りでスキーマを記述し、何かで検査しなければ両者はずれていきます。この例では、データベースを1,022本のマイグレーションで版管理し、それぞれに036から1102までの番号を付けています。変更のたびにCIが二度構築し、一度は新規に、もう一度はすべてのマイグレーションを順に再実行して、構造だけでなく初期データも比較します。参照テーブルの補充を間違えるマイグレーションは、列を追加し忘れるものと同じだけ損害を与えます

が、スキーマの差分に現れるのは一方だけです。食い違いがあれば、差分を表示してビルドが失敗するため、すべてのマイグレーションは、再実行でも初期構築からでも機能しなければなりません。

問題が起きるところ

共有スキーマに、関数を仕込まれる。 PostgreSQL wikiのCVE-2018-1058対策ガイドは、その攻撃を示しています。ある利用者が、publicに `lower(varchar)` を作ります。組み込みの `lower` はpg_catalogにあり、textを取るため、varcharの列では仕込んだ関数の方が適合します。同僚が `SELECT lower(full_name) FROM a` を実行すると、代わりに仕込まれたコードが動きます。重要なコードでは `pg_catalog.lower` のようにスキーマを付け、必要のないロールからpublicのCREATE権限を外し、 `\df public.*` で、すでに存在するものを列挙します。

更新しても、古い権限が残る。 PostgreSQL 15の安全性を高めた既定値が適用されるのは、新しいクラスターと、新規に作成したデータベースです。クラスターの更新やダンプからの復元は、publicの既存の権限を保つため、15以降のサーバーでも、すべてのロールがそこにオブジェクトを作れる場合があります。マニュアルは、publicにCREATEを許可した既定の検索経路を、決して安全な使い方ではないとし、14以前から更新したデータベースの既定値だと説明しています。そのようなデータベースでは、 `REVOKE CREATE ON SCHEMA public FROM PUBLIC` を実行し、pg_catalog内と同じ名前のオブジェクトがpublicにないか調べます。

テーブルを間違った場所に作る。 マイグレーションが、スキーマなしで `invoices` を作り、実行するロールの検索経路で、最初に存在するスキーマへ入ります。別の経路で接続するアプリケーションは、違うテーブルを読むか、 `relation "invoices" does not exist` を受け取ります。マイグレーションではすべての名前にスキーマを付け、実行するセッションで `SELECT current_schema()` を確認します。

削除が、スキーマの外まで及ぶ。 `DROP SCHEMA billing` は、RESTRICTが既定なので、中に何かあれば削除を拒否します。CASCADEを付けると、内部のすべてのオブジェクトと、それらに依存するすべてのオブジェクトを削除します。マニュアルは、指定したものとは別のスキーマのオブジェクトも削除し得ると警告しています。まずCASCADEなしでDROPを実行し、削除対象を列挙するDETAIL行を読むよう勧めています。

システムが先に解決する名前。 pg_catalogは、search_pathに含まれていない場合は、一覧のスキーマより先に暗黙的に検索されます。明示的に含めると、経路内の位置が検索順を決めます。自作のテーブル名が `pg_` で始まると、後のバー

ジョンで同名のシステムテーブルが加わった際に隠され得るため、マニュアルは避けるよう勧めています。`user` や `order` などの予約語も、程度は軽いものの、同じ問題を起こします。そのようなテーブル名は、すべての文で二重引用符が必要です。

参考資料

- 5.10. スキーマ — スキーマを付けた名前、publicスキーマ、search_path、安全な使い方、移植性を扱うマニュアルの章です。
- CREATE SCHEMA — コマンドの構文、AUTHORIZATION句、予約済みの `pg_` 接頭辞、PostgreSQLが標準と異なる点です。
- 5.8. 権限 — スキーマのUSAGEとCREATEの意味を、他の権限とともに説明します。
- E.20. リリース15 — 新しいデータベースではpublicのCREATEをなくし、更新したものでは保ったリリースです。
- CVE-2018-1058ガイド：検索経路の保護 — 仕込んだ関数による攻撃を、対策と、データベースを監査するコマンドとともに順に説明します。

用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「スキーマとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/schema/