Engineer

PostgreSQL

ここにある仕事の大半で基盤となったデータベース

どのようなものか

PostgreSQLは、無償で利用できるオープンソースのリレーショナルデータベースサーバーです。データを表に保存し、SQLで書いた問い合わせに答え、各トランザクションを、すべての変更を確定するか、すべて取り消すかの単位として扱います。使っている人の多くはPostgresと呼び、プロジェクトも両方を自身の名前としています。「PostgreSQLとは何か」と現場のエンジニアに聞けば、ほかを選ぶ具体的な理由がないとき、新しいシステムの出発点にする汎用データベース、と答えるでしょう。

現在その上で動くソフトウェアの大半よりも、長い歴史があります。POSTGRESプロジェクトは、カリフォルニア大学バークレー校でMichael Stonebraker教授の下に始まり、1986年に実装が始まりました。Andrew YuとJolly Chenが1994年にSQLインタープリターを加え、Postgres95として公開しました。その二年後、PostgreSQLへ改名し、バークレーで始めた番号の系列を引き継ぐため、バージョン番号を6.0から再開しました。プロジェクト自身は、システムをオブジェクトリレーショナルと説明しています。表や結合だけでなく、新しいデータ型、関数、演算子、集約関数、索引方式、手続き言語を利用者が追加でき、サーバーはそれらを自身の機能と同じように扱います。

構造はクライアントとサーバーに分かれます。一つのサーバープロセスがデータファイルを管理し、クライアント接続ごとに、その接続専用にforkしたプロセスを割り当てます。アプリケーションは、実装言語に対応するドライバーを通じ、その接続でSQLを送ります。同時実行の基礎は多版型同時実行制御です。更新では既存の行を上書きせず、新しい版を作ります。そのため、各文は整合したスナップショットを読み、読み取り側が書き込み側を待つことも、書き込み側が読み取り側を待つこともありません。

ほかの選択肢との位置関係はどうでしょうか。オープンソースのリレーショナルサーバーとしてMySQLやMariaDBと並び、商用サーバーとしてはOracleやMicrosoft SQL Serverがあり、一つのプログラム内で動く組み込みデータベースとしてSQLiteがあります。固定スキーマなしでレコードを保存するMongoDBのようなドキュメントストアとも比較されます。

概念を判断に生かす

「PostgreSQL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。 年々重くなってきたシステムのクエリと不足するインデックスを調べ、効果のある箇所を改善します。

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。 30日分の電力データを180 msで処理 — 180 ms 1か月分のメーター値を集計

30日分の電力データを180 msで処理 お問い合わせ →

なぜ重要なのか

データベースを選ぶと、その判断と何年も付き合うことになります。データは、それを使うために書いたコードより長く残るからです。PostgreSQLを標準候補にして後悔することが少ない理由は三つあり、購入を検討する側でも確かめられます。

一つ目は費用とベンダーへの依存です。BSDやMITと同じ系統の、短く、利用条件の緩やかなライセンスであるPostgreSQL Licenseで公開されます。コアごとの料金も監査もなく、条件を変更できるベンダーもいません。Amazon、Microsoft、Googleがそれぞれマネージドサービスとして販売し、同じデータベースがノートパソコンでも動きます。そのため、ホストを移す際に必要なのはデータの移行であり、アプリケーションの書き直しではありません。

二つ目は扱える範囲です。一つのサーバーで、リレーショナルな表、`jsonb`列内のJSON文書、全文検索、PostGIS拡張による地理空間クエリー、`pg_cron`による定期処理を扱えます。それぞれを別製品にすれば必要になる運用、安全確保、バックアップ、人材の採用を、チームは抱えずに済みます。多くの製品では、PostgreSQLかMySQLかの判断はここで決まります。単純な読み書きならどちらも十分に使えますが、より豊かな型、厳密な制約、拡張機能が必要になると違いが表れます。

三つ目は長く使えることです。プロジェクトは毎年一つのメジャーバージョンを公開し、それぞれを五年間サポートします。緊急事態に迫られて更新するのではなく、計画して更新できます。2025年9月公開のPostgreSQL 18では、非同期I/O

と、時刻順のUUIDを作る組み込み関数が加わりました。十年前の版向けに書かれたSQLも、通常は変更せずに動きます。

運用上の負担もあります。接続ごとにプロセスが必要なため、接続のコストは高く、負荷の大きいアプリケーションには接続プールを管理する仕組みが必要です。複数のマシンに広げるより、一台を大きくする方が容易です。書き込みを多くのサーバーへ分散するには、パーティション分割、レプリケーション、または拡張機能が必要で、いずれも運用の負担を増やします。また、既定値は小規模なマシン向けなので、本格的な環境では意図して設定を調整する必要があります。

実際の活用例

このサイトにあるプロジェクトの多くはPostgreSQLで動いています。そのうち三つが、異なる三通りの使い方を示します。

あるプラットフォームでは、データベースがデータ層のすべてを担います。読み書きはすべて、34個のスキーマにまたがる1,275個のPostgreSQL関数のいずれかを通り、GoのAPIはどの表にも直接アクセスできません。APIがログインに使う `mariner` ロールには、関数に対するEXECUTEとスキーマに対するUSAGEだけを与え、明示的な権限付与は約4,000件です。関数は、ログインできない別の所有者の下で、 `search_path` を固定してSECURITY DEFINERとして動きます。アクセスの境界はPostgres自身が守りますが、フィールドを追加するにはマイグレーションと関数の変更が必要になります。

テナント、場所、日ごとの電力検針値を保持するアプリケーションでは、設計ではなく性能調整が仕事でした。求められたのは、構造を全面的に作り直さずに性能を10倍にすることです。空間データ用のGiST索引、テナントと日付範囲の複合索引、電力テーブルの日・月単位の範囲パーティション分割、 `EXPLAIN ANALYZE` で計画を読んでからのクエリー書き換えにより、クエリーの応答時間を10分の1に短縮し、リソース使用量を40%削減しました。

また、時間のかかるバックグラウンド処理があるプラットフォームでは、ジョブキューを同じデータベースに置きます。RiverはキューをPostgreSQLに保存するため、ジョブと、そのジョブが触る行は、まとめてコミットまたはロールバックされます。その背後には、15個のワーカーモジュール、8個の定期タスク、20個の `pg_cron` ジョブがあり、古いパーティションを取り除き、ソルトを更新し、集約値を再計算します。ページは限界も明記しています。ある処理量を超えればPostgres内のキューは適さなくなりますが、誰も使わない余力を確保するより、運用する部品を一つ減らす方に価値がありました。

問題が起きるところ

PostgreSQLでデータが失われることはまれです。一方、使う人が驚くことはあり、その原因にはよくある形があります。

行数が変わらないのに表が大きくなります。 `UPDATE` や `DELETE` の後には、別のトランザクションがまだ読んでいる可能性があるため、`VACUUM` が取り除くまで古い行の版が残ります。何時間も開いたままのトランザクションがあると、データベース全体でその掃除が進まなくなります。autovacuumの既定値も、小さい表向けです。アイドル状態のトランザクションを閉じ、大きい表では表ごとにautovacuumを調整し、期限が切れるデータを分割して、古い月の行を削除するのではなく、月のパーティションを丸ごと落とします。vacuumにはもう一つ役目があります。トランザクションIDは32ビットで、各表は二十億トランザクションごとに少なくとも一度vacuumする必要があります。処理が遅れすぎたサーバーは、データを守るため新しい書き込みを拒否します。

最初のアクセス集中でメモリーが尽きます。 各接続は自身のメモリーを持つサーバープロセスで、既定の上限は100接続です。アプリケーション二十インスタンスが各五十接続のプールを持てば、千プロセスを要求します。PgBouncerなどの接続プーラーを前に置き、各アプリケーションのプールを処理に見合う大きさにします。通常は、一インスタンスあたり数接続で足ります。

ミリ秒で終わっていたクエリーが、秒単位になります。 データが増えるにつれてプランナーが別の計画を選んだためで、多くの場合、統計が古くなったか、索引が一年前のクエリー向けに作られたことが原因です。変更する前に計画を読み、`ANALYZE` で統計を更新し、実際に絞り込みに使う条件に索引を付けます。

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, total FROM orders
WHERE customer_id = 42 AND placed_at >= '2026-01-01';
```

表が大きくなると挿入が遅くなります。 ランダムなUUIDを主キーにすると、B-tree索引の各所に入るため、挿入のたびに異なるページへ触れ、索引が断片化します。時刻順の識別子なら、この散らばりを避けられます。あるプラットフォームでは、PostgreSQL 18の `uuidv7()` でUUID v7を既定のキーにしました。新しい行は索引内で順に並び、あちこちに散らばりません。

参考資料

- PostgreSQLとは — プロジェクト自身による定義で、機能と拡張できる箇所を挙げています。
- PostgreSQLの略史 — Berkeley、Postgres95、改名について、実際に関わった人たちが説明しています。
- 同時実行制御：概要 — スナップショットにより、読み書きが互いを妨げずに進む仕組みです。
- 定期的なvacuum — 不要な行の版、autovacuum、トランザクションIDの周回を詳しく扱います。
- EXPLAINの使い方 — 実例を使ったクエリー計画の読み方です。
- バージョン方針 — 年一回のメジャーリリースと、各版の五年間のサポートを説明しています。

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

「PostgreSQL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/postgresql/