Engineer

パーティショニング

一つのテーブルを物理的な部分に分けること

どのようなものか

パーティショニングは、テーブルパーティショニングとも呼び、一つの論理テーブルの行を、複数の通常のテーブルに保存する方法です。各テーブルには、その範囲にキーが入る行を置き、クエリーや書き込みは、従来と同じ一つの名前を使い続けます。このページで取り上げるPostgreSQLは、組み込みの方式を宣言的パーティショニングと呼びます。方式とパーティションキーを指定してテーブルを分割対象と宣言し、各パーティションを、それぞれ独立したテーブルとして作ります。

パーティション化された親テーブル自身は、行を持ちません。挿入は、キーが範囲に合うパーティションへ振り分け、キーを変える更新では、新しい値の所属先へ行を移します。パーティションは親と列を共有しますが、独自のインデックス、ストレージ設定、テーブルスペースを持てます。パーティションを、さらにパーティショニングすることもできます。

方式は三つあります。範囲による方式は、各パーティションに、通常は日付などの値の区間を割り当てます。下限を含み、上限を含まないため、1–10と10–20なら、10は後者に入ります。リストによる方式は、国コードの集合など、各パーティションが受け入れる値を列挙します。ハッシュによる方式は、各パーティションに法と剰余を与え、キーのハッシュをその法で割った余りが、指定の剰余と一致するパーティションへ行を振り分けます。自然に区切れる範囲がないときも、行を均等に配れます。どこにも合わない値はデフォルトパーティションに収められますが、使えるのは範囲方式とリスト方式だけです。

パーティショニングは、一台のデータベースサーバー内で行います。スキーマのコピーをそれぞれ持つ別々のサーバーへ分けるのはシャーディングで、はるかに大きな変更です。SQL ServerとBigQueryにも、独自の構文による方式があり、コマンドが違っても、以下の考え方は当てはまります。

概念を判断に生かす

「パーティショニングとは？」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

大きなテーブルの多くは、一方の端から増えます。注文、測定値、ログ、イベントが日々入り、数か月や数年残り、クエリーの多くは最近の部分を求めます。分割しなければ、一つのテーブルと、増え続ける一組のインデックスを、すべてのクエリーが相手にします。

クエリーに合うパーティションキーを選ぶと、プランナーは、多くのパーティションに該当行が入り得ないと証明し、計画から丸ごと外せます。PostgreSQLは、これをパーティションプルーニングと呼びます。先週のレポートは、先週のパーティションだけを読みます。ドキュメントは、見えにくい利点も挙げます。パーティションがインデックスツリーの上位層の役割を担うため、よく使うインデックスの部分が、メモリーに収まりやすくなります。

最も変わるのは、古いデータの削除です。一つのテーブルから一か月分を消すと、長い `DELETE` になり、残った不要な行を `VACUUM` で掃除しなければなりません。月ごとに分ければ、同じ保持方針を、一回の `DROP TABLE` か `DETACH PARTITION` で実行できます。ドキュメントは、大量の行をまとめて操作するよりはるかに速く、 `VACUUM` の負担も全くないと説明します。切り離しても通常のテーブルなので、削除前にバックアップや集計を行えます。ほとんど読まないパーティションは、安価で遅いストレージにも置けます。

利点には代償があり、ドキュメントは割に合う条件も明記します。パーティションごとに計画の仕事が増えるため、通常は、分割しなければ非常に大きくなるテーブルで初めて利点が見合います。目安は、データベースサーバーの物理メモリーより大きいことです。変わらない点もあります。多数のトランザクションが奪い合う一行は、分割後も、一つのパーティション内で、一つのロックに守られた一行のままです。

実際の活用例

最も単純な例は、`orders` を `created_at` で月ごとに分ける形です。シャーディング前の拡張方法を扱うノートが、競合の激しい一行の周りでテーブルを分ける方法に一節を割いています。主キーは、分割列を含める必要があるため、`(id, created_at)` です。日付範囲を指定するクエリーは、その中のパーティションだけを読み、今月のインデックスはメモリーに収まる大きさに保たれます。保持期間を過ぎたデータは、一回の `DETACH PARTITION` で外せます。数百万行を `DELETE` する代わりです。限界も明確です。競合する一行を速くすることはなく、キーを指定しないクエリーは、すべてのパーティションを読みます。

利用者アカウント、テナント、地理データ、日々の電力消費を扱うアプリケーションのPostgreSQL性能改善では、電力のテーブルを日単位と月単位の時間範囲に分割しました。地理データにはハッシュによる分割を使いました。GiSTと複合インデックス、クエリーの書き換えも組み合わせ、その総合的な結果として、クエリーの応答時間が10分の1に短縮し、資源の使用率が40%下がりました。

パーティショニングは、はるかに大きな規模では、PostgreSQL、SQL Server、BigQueryにまたがる少なくとも100のデータベースに使った、複数の手法の一つでした。多くが地理データと時系列データで、PostgreSQLとSQL Serverでは、時系列の分割をスキーマ設計に含め、BigQueryでは時間で分割してクラスタリングしたテーブルを使いました。空間インデックス、マテリアライズドビュー、キャッシュと合わせて、そのデータへのクエリー応答時間が40–60%短くなりました。

問題が起きるところ

パーティショニングは、仕事の一部をクエリー実行時から設計時へ移します。問題の多くは、実際のクエリーを考えずにした設計判断から生まれます。

月初に挿入が失敗し始めます。 新しい範囲を受け持つパーティションがなく、既存のどれにも対応しない親への挿入を、PostgreSQLが拒否します。ドキュメントの提案どおり、DDLを生成する定期ジョブで、前もってパーティションを作ります。最新のものが範囲の終わりに近づいたら、アラートを出します。

```
CREATE TABLE orders_2026_11
  PARTITION OF orders
  FOR VALUES FROM ('2026-11-01')
              TO ('2026-12-01');
```

デフォルトパーティションがあればエラーは止まりますが、受け皿のない全月の行が、一つのテーブルに集まります。後で新しいパーティションを付けると、`ACCESS EXCLUSIVE` ロックの下でデフォルト側を走査します。ただし、デフォルト側に、新しい範囲をあらかじめ除外する `CHECK` 制約があれば、走査は不要です。デフォルトが空でなければ、警告として扱います。

分割後の方が、クエリーが遅くなります。 通常のクエリーがパーティションキーで絞らず、すべてのパーティションとインデックスを読むためです。数の増え過ぎも原因になります。ドキュメントは、通常のクエリーが数個以外を除外するなら、数千程度までは十分扱える一方、それを超えると計画の時間とメモリーが増えるとします。アプリケーションが実際に書く `WHERE` 句からキーを選び、`EXPLAIN` で、想定したパーティションだけが現れることを確かめ、数を適度に保ちます。顧客ごとに一つ作ると、事業とともに増える、典型的な落とし穴になります。代わりに、妥当な数のハッシュパーティションを使うよう、ドキュメントは勧めます。

主キーや一意制約を作れません。 分割したテーブルでは、一意キーや主キーに、パーティションキーの全列を含める必要があり、分割キーに式や関数呼び出しは使えません。各パーティションのインデックスが一意性を保証できるのは、その内部だけです。ノートの `orders` が `(id, created_at)` とするように、制約にキー列を加えます。

親のインデックス作成が、長時間書き込みを止めます。 親で通常の `CREATE INDEX` を実行すると、全パーティションのインデックス作成が終わるまで書けません。`CONCURRENTLY` は親ではなく、個々のパーティションで使います。親には `ON ONLY` でインデックスを作り、無効の印を付けたまま、各パーティションのインデックスを `CONCURRENTLY` で作り、`ALTER INDEX ... ATTACH PARTITION` で一つずつ結び付けます。すべてが結び付けば、親のインデックスも自動的に有効になります。

既存の大きなテーブルを、そのまま切り替えられません。 PostgreSQLは、通常のテーブルを分割テーブルには変えません。新しい分割テーブルを作り、古いテーブルを、その一つのパーティションとして取り付けます。先に、パーティションの範囲に合う `CHECK` 制約を加えると、取り付け時に `ACCESS EXCLUSIVE` ロックの下で行う、テーブル全体の走査を避けられます。

参考資料

- テーブルのパーティショニング — PostgreSQLの章で、三つの方式、プルーニング、保守、制約、推奨事項を説明します。

- CREATE TABLE — `PARTITION BY` 、 `PARTITION OF` 、パーティションの範囲、デフォルトパーティションです。
- ALTER TABLE — `ATTACH PARTITION` と `DETACH PARTITION` 、それぞれが取るロックです。
- CREATE INDEX — 分割テーブルのインデックス、 `ON ONLY` 、 `CONCURRENTLY` での作成です。

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

「パーティショニングとは？」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/partitioning/