Engineer

マテリアライズドビュー

クエリー結果をテーブルのように保存するビュー

どのようなものか

マテリアライズドビューは、データベースが結果を保存するクエリーです。読むときは再実行せず、以前計算した行を返します。英語の英国式綴りはmaterialised viewです。古くからある二つのオブジェクトの中間にあります。通常のビューは、クエリーだけを保存し、読むたびに元のテーブルに対して実行します。`CREATE TABLE AS`で作るテーブルは、結果を保存しますが、それを作ったクエリーは忘れます。マテリアライズドビューは、読み取り元の行と、再実行して行を置き換えられるクエリーの両方を保ちます。

PostgreSQLは、2013年9月9日公開の9.3で、マテリアライズドビューを追加しました。リリースノートは、この仕事をKevin Grittnerによるものとしています。文書の説明では、通常のビューと同じようにルールシステムを使いながら、テーブルに似た形で結果を永続化します。クエリー解析器にとっては、テーブルやビューと同じリレーションなので、`SELECT`は保存済みの行を直接読みます。保存したクエリーが関係するのは、内容を入れるときだけです。ビューを直接更新することはできません。内容を新たなクエリー結果に置き換えるには、保存したクエリーを再実行する必要があります。

その仕事をするコマンドは二つです。`CREATE MATERIALIZED VIEW`は、ビューを定義し、`WITH NO DATA`の指定がなければ、すぐにクエリーを実行して内容を入れます。`REFRESH MATERIALIZED VIEW`は、保存したクエリーを再実行し、内容全体を新しい結果に置き換えます。2014年12月18日公開の9.4では、読み取り中のセッションを止めずに更新する`CONCURRENTLY`を追加しました。両コマンドはPostgreSQLの拡張であり、SQL標準には含まれません。他のデータベースは、独自の構文と規則で同じ考えを提供します。

概念を判断に生かす

「マテリアライズドビューとは？」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

計算が重く、何度も求められるクエリーがあります。一年の注文を日別に合計するダッシュボード、六つのテーブルを結合する報告、売上順に商品を並べるページなどです。元のテーブルに実行すれば、答えがほとんど変わらなくても、各リクエストで全走査、結合、集計の処理コストを繰り返し払います。マテリアライズドビューなら、再計算のたびに一度払い、その後の読み取りは、事前に作った小さなテーブルの走査になります。PostgreSQLの文書は、保存したデータを読む方が、「元のテーブルを直接、またはビュー経由で参照するより、はるかに速いことが多い」と述べます。

代価は、鮮度です。再計算の間、ビューは前回時点のデータを示します。PostgreSQLでは、人かスケジュールされた処理が要求しなければ、更新されません。多くの用途は、それを許容できます。昨日の売上は今日変わらず、月次報告には、この一分間に入った注文は不要です。マテリアライズドビューが求めるのは、答えがどれだけ古くてもよいか、それより古くさせない責任を誰が持つかという、明示的な判断です。

インデックスを置く場所にもなります。保存した結果には、それを読むクエリー向けの独自のインデックスを作れます。PostgreSQLの文書は、別の使い道も示します。外部データラッパーを通じて別のシステムから得たデータは、そのラッパーがインデックスにまったく対応しない場合でも、インデックスを持てるマテリアライズドビューにコピーできます。

スキーマを正規化して保つチームでは、平坦にまとめた形を置く場所として、よく適しています。正規化は、各事実を一度だけ保存し、大半の問いに結合で答えます。マテリアライズドビューに、報告で必要な結合済み、集計済みの結果を持たせれば、元のテーブルは、各事実の単一のコピーを保ち、書き込みも一つの場所に向けられます。

実際の活用例

店の注文テーブルを考えます。`orders`は、注文ごとに一行あり、そこに注文日を記録し、`order_lines`は、各明細のSKU、数量、単価を持ちます。売上ダッシュボードが求めるのは、商品ごと、日ごとの数量と売上高です。クエリーは短くても、何年分もの注文を対象にすれば、すべての明細を読みます。

```
CREATE MATERIALIZED VIEW daily_sales AS
SELECT o.ordered_on                  AS day,
       l.sku,
       sum(l.quantity)               AS units,
       sum(l.quantity * l.unit_price) AS revenue
FROM   orders o
JOIN   order_lines l USING (order_id)
WHERE  o.ordered_on < CURRENT_DATE
GROUP  BY o.ordered_on, l.sku;
```

`WHERE`は、まだ数値が変わる今日を外すため、ダッシュボードには、終わった日だけが載ります。PostgreSQL自身の文書の例も、売上の確定していない一日を同じように扱います。コマンドは、クエリーを一度実行し、結果を保存します。続いてインデックスを作ります。

```
CREATE UNIQUE INDEX daily_sales_sku_day ON daily_sales (sku, day);
```

インデックスには、二つの役割があります。ビュー全体を読まずに、一商品の各日を見つけられるようにし、後から、読み取りと並行して再計算することも可能にします。クエリーは日とSKUでまとめるため、組み合わせは一度ずつ現れ、一意にできます。その後は、ビューをテーブルと同じように読みます。

```
SELECT day, units, revenue
FROM   daily_sales
WHERE  sku = 'TEA-250'
ORDER  BY day;
```

毎晩、真夜中を少し過ぎたころ、定期実行するジョブが最新にします。

```
REFRESH MATERIALIZED VIEW CONCURRENTLY daily_sales;
```

`CONCURRENTLY` なしでは、多数の行に触れる再計算は、使う資源が少なく、早く終わる傾向がありますが、終わるまで読み取りを止める可能性があります。指定すれば、更新中も読めます。文書によれば、ここのように毎晩一日分が加わるだけで、変わる行が少ないなら、こちらの方が速い場合さえあります。条件は二つあります。既にデータが入っていることと、式ではなく列名そのものから作り、すべての行を覆う一意インデックスが少なくとも一つあることです。式インデックスも、 `WHERE` 付きも認められません。上のインデックスが二つ目を、データを入れてビューを作ったことが一つ目を満たします。

他のデータベースは、線を引く場所が違います。BigQueryは、マテリアライズドビューをバックグラウンドで維持します。元テーブルの変更を、要求なしに増分で取り込み、ビューへのクエリーでも最近の変更を合わせ、結果を最新にします。スマートチューニングは、元テーブル向けのクエリーを、ビューに振り向けることもできます。これは、SQLの一部だけを受け入れる増分ビューの場合です。 `allow_non_incremental_definition = true` で定義したビューは、大半のクエリーを受け入れますが、更新のたびに全クエリーを実行し、結果も常に最新とは限りません。

SQL Serverは、インデックス付きビューで似た結果を得ます。 `WITH SCHEMABINDING` で作成し、定義が決定的であるビューに、一意性を強制するクラスター化インデックスを加えると、インデックス付きテーブルと同じ格納構造で結果を保存します。最適化器は、ビューを名指ししないクエリーにも使えます。SQL Serverは、依存するテーブルで挿入、更新、削除するたびに、ビューも更新するため、再計算の予定は不要です。処理コストは書き込み側に移り、Microsoftの文書は、書き込み性能が大幅に落ちる可能性を警告します。

本番の仕事にも、同じ利点と代償が現れます。創業者は、以前の職務でPL/pgSQLのETLとELTのパイプラインを構築しました。大きなデータ集合の読み取りを速くする、パーティション化したテーブルとマテリアライズドビューに加え、参照整合性のための主キー、外部キー制約と、結合用の一意インデックス、複合インデックスを使いました。

問題が起きるところ

古い数値が、最新に見えます。 一週間前に更新ジョブが止まっても、ビューは全クエリーに、速く、間違った答えを返し、結果は古さを示しません。成功した更新の時刻を、ダッシュボードが表示できる場所にジョブに記録させ、読み手が合意した許容時間を超えたら通知します。

再計算が、読み手を止めます。 大きなビューへの通常の `REFRESH MATERIALIZED VIEW` は、終わるまで、読もうとするすべてのセッションを止める可能性があります。一意インデックスを足して `CONCURRENTLY` を使うか、誰も読まない時間に通常の更新を行います。毎回、大半の行が変わるなら、通常の更新の方が安い場合もあります。

`CONCURRENTLY` **が拒否されます。** 式ではなく列名そのものを使い、全行を覆う一意インデックスが必要です。 `lower(email)` の一意インデックスや、 `WHERE` で範囲を限るものは認められません。既にデータがあることも必要なので、 `WITH NO DATA` で作ったビューでは失敗し、両方の指定を同時にも使えません。条件を満たすインデックスを加え、一度、通常の更新でデータを入れます。

走査できないビューです。 `WITH NO DATA` は、クエリーを実行せずにビューを作り、走査不可とするため、最初の再計算までは、どの読み取りも失敗します。長いクエリーを待てない移行には適することがありますが、その場合、最初の再計算も忘れずに計画する必要があります。

定義の変更が届きません。 移行の `CREATE MATERIALIZED VIEW IF NOT EXISTS` は、同名のビューが既にあれば、通知以外は何もしません。文書は、既存のビューが新しい定義に似ていることさえ保証しません。この方法でクエリーを変えても、古いものが残ります。同じ移行内で、 `DROP MATERIALIZED VIEW` で削除して作り直します。

更新ジョブが重なります。 `CONCURRENTLY` でも、一つのマテリアライズドビューで進められる再計算は一つだけです。再計算の所要時間より間隔を短くしても、さらに遅れるだけです。所要時間を測り、それを基に予定します。

行の順序を信用します。 定義のクエリーに `ORDER BY` があれば、最初の内容はその順で出ますが、更新後も保つ保証はありません。読む側のクエリーに `ORDER BY` を置きます。

関数が、自分のテーブルを見失います。 作成時でも更新時でも、定義のクエリーを実行する間、PostgreSQLは `search_path` を一時的に `pg_catalog, pg_temp` とし、できる処理も制限します。一時テーブルを作る関数は失敗します。テーブルをスキーマ名なしで探す関数は、手動テストをすべて通っても、再計算中だけ壊れる場合があります。ビューが呼ぶ関数では、テーブル名にスキーマを付けます。

違うロールでジョブを動かします。 更新には、マテリアライズドビューの `MAINTAIN` 権限が必要です。読むことだけ許され、保守は許されない報告用ロールで動くジョブは、毎晩失敗します。更新ジョブのロールに、その権限を与えます。

インデックスで済むのに、コピーを保存します。 インデックスがないために遅いクエリーでは、マテリアライズドビューは問題を隠し、データの古さという問題まで加えます。先にクエリープランを読みます。名前を付けるだけなら、常に最新の通常のビューで十分です。マテリアライズドビューは、計算が重く、少し古くても安全に読める結果のために残します。

参考資料

- マテリアライズドビュー — ビューやテーブルとの関係を扱うPostgreSQLの章です。毎晩更新する売上集計と、外部データラッパーの例があります。
- CREATE MATERIALIZED VIEW — `WITH NO DATA` 、 `IF NOT EXISTS` 、実行中の制限付き `search_path` を含む、コマンドの参照資料です。
- REFRESH MATERIALIZED VIEW — 更新コマンド、 `CONCURRENTLY` の条件、必要な `MAINTAIN` 権限です。
- PostgreSQL 9.3のリリースノート — 2013年9月、マテリアライズドビューを追加した版です。
- マテリアライズドビューの概要 — BigQueryの増分と非増分のビュー、自動更新、スマートチューニングです。
- インデックス付きビューを作る — SQL Serverのインデックス付きビュー、条件、書き込みにかかる処理コストです。

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

「マテリアライズドビューとは？」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/materialized-view/