Engineer

クエリー

▼ データベースに答えを求めて書く問い合わせ

どのようなものか

クエリーは、データベースが理解する言語で書かれたデータへの要求で、結果が返ります。通常は行の集合で、ときには一つの数値です。PostgreSQLのマニュアルは、一文で定義しています。データベースからデータを取得すること、または取得するコマンドをクエリーと呼び、SQLでは、そのコマンドがSELECTです。日常的な使い方では意味が広がり、エンジニアは、データを変更して何かを返す文を含め、データベースへ送るほぼすべての文をクエリーと呼びます。以下には、両方の意味が出てきます。複数形のqueriesは、通常、アプリケーションが送るすべての通信を指します。

どの請求書が最も長く支払い待ちになっているかを尋ねるクエリーは、口頭の質問に近い表現になっています。

```
SELECT i.number, c.name, i.amount, i.due_on
FROM invoices AS i
JOIN customers AS c ON c.id = i.customer_id
WHERE i.paid_on IS NULL
  AND i.due_on < current_date
ORDER BY i.due_on
LIMIT 20;
```

記述するのは、何が欲しいかで、取得する方法は指定しません。PostgreSQLのプランナーは、受け取ってから最初の行を返すまでの間に、請求書のテーブル全体を読む、`due_on` の索引をたどる、といった答えを得る方法を列挙し、それぞれのコストを見積もり、コストが最も小さい方法を実行します。そのコストは、慣例としてディスクから一ページを読むコストを基準とする、任意の単位で表します。そのため、画面上では似て見える二つのクエリーでも、対象データが増えると、コストが千倍違うことがあります。違いが見えるのは、実行計画だけです。

各句は、書かれた順番では実行されません。マニュアルが説明するSELECTの処理順は、最初にWITH、次にFROM、WHERE、グループ化とHAVING、その次に出力する列の一覧で、ORDER BYとLIMITは、その後です。多くの人が一度は戸惑う規則も、この順番で説明できます。SELECTの出力列に付けた名前は、ORDER BYとGROUP BYでは使えますが、WHEREは、その名前が存在する前に処理されるため、式をもう一度書く必要があります。どのクエリーも結果自体がテーブルになるため、入れ子にでき、サブクエリーやWITH句の行を、より大きいクエリーへ渡せます。

概念を判断に生かす

「クエリーとは？」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

リレーショナルデータベースを使うアプリケーションでは、ページの体感速度や、システムに必要なハードウェアの量が、クエリーに大きく左右されます。どのクエリーが重要かは、直感ではなく計算で分かります。システムにとってのコストは、一回にかかる時間に実行回数を掛けた値です。2 msの検索も、一時間に百万回実行すれば、毎時、合計で三十分を超えるデータベースの処理時間になります。一方、二秒かかるレポートでも、一日に二回なら、ほとんど影響しません。

PostgreSQLは、pg_stat_statements拡張で、その集計を行います。リテラルの値だけが違う文を一つの項目にまとめ、それぞれの実行回数と合計時間を記録します。平均ではなく合計で並べることが、クエリーの改善で最初に役立つ習慣です。遅く見えるだけの文ではなく、直す価値がある文を指し示すからです。

どのクエリーにも、往復通信のコストがあります。アプリケーションが文を送り、サーバーが構文を解析し、計画して実行し、行が戻ってきます。索引を使う速い検索では、処理そのものより、往復の方が重い場合があります。そのため、個々の文と同じくらい、やり取りの形が重要です。百行を変更する一つのUPDATEなら、往復は一回です。百行を読み、一行ずつ更新するループでは、百一回になります。

クエリーを誰が書くか、という問題もあります。ORMやクエリービルダーは、アプリケーションのコードから生成するため、単純な読み書きを手早く実装でき、コードを一つの言語に保てます。一方、ページが遅くなるまで、一つのリクエストが送る文の数や内容を隠してしまいます。手書きのSQLなら、両方が見えますが、ほかのコードと同様に、誰かが責任を持って保守する必要があります。実用的な分担は、日常的なレコード操作をORMに任せ、性能上重要な処理経路を手書きし、合計コストの上位に来る処理では、生成されたSQLを読むことです。

実際の活用例

データベースを分割する前に最適化するで示す、遅いデータベースへの対応は、誰かが追加マシンの価格を調べるより先に、クエリーから始めます。最初の手順では、pg_stat_statementsを合計実行時間で並べ、全体で最もコストのかかる文を、PostgreSQLに尋ねます。次に、一覧の先頭の文にEXPLAIN (ANALYZE, BUFFERS)を実行し、選ばれた計画、各手順が実際に返した行数とプランナーの予測との差、読んだページ数を確認します。このノートの目安は、十行を返すために百万行を走査している場合や、見積もりが百倍外れている場合は、それだけで遅さを説明できることが多い、というものです。往復回数も数え、この記事と同じ境界を引きます。データへアクセスする必要がある処理はデータベースへ移し、業務規則はアプリケーションに残します。

クエリーは、ロックの役割も果たせます。最後の一枚を一度だけ売るでは、短時間に注文が集中する販売で、二種類の失敗が起きます。在庫を読み、アプリケーションで判断し、新しい数を書き戻す方法では、二人の購入者が、どちらも最後の一個を取れてしまいます。そのアプリケーションの処理中ずっと行ロックを保つと、全購入者が、その後ろで待つことになります。解決策は、在庫の確認と減算を、一つの条件付きUPDATEへまとめることです。十分な在庫がある場合だけ数量を引き、新しい在庫数を返します。一行返れば確保でき、零行なら売り切れです。PostgreSQLの既定の分離レベルであるREAD COMMITTEDでは、一人目が行を保持している間に二人目が到達すると、二人目は待ち、コミット済みの行に対してWHERE句を再評価します。そのため、最後の一個を得るのは、必ずどちらか一人です。在庫をゼロ以上に保つCHECK制約が、このクエリーを補強します。

問題が起きるところ

軽いクエリーが、負荷の大半を占めます。 データベースの負荷は高いのに、各呼び出しは数ミリ秒で終わるため、遅いクエリーのログには何も出ません。原因は頻度です。リクエストのたびに、一つの速いクエリーを呼び、多くの場合、ル

ープや、全ページに現れる部品から実行しています。pg_stat_statementsで文を合計実行時間順に並べ、結果をキャッシュする、呼び出しをまとめる、絞り込み条件に合う索引でクエリーの処理負担を下げる、といった対策を取ります。

一ページが、何百本ものクエリーを送ります。 トレース内の各クエリーは一ミリ秒でも、ページには数秒かかります。典型的な原因は、ORMが、親の行ごとに一つのクエリーを送り、関連するレコードを遅延読み込みすることで、N+1パターンと呼ばれます。結合か、`WHERE id = ANY($1)` とキーの配列を使い、関連する行を一つのクエリーで読みます。一つの文の実行時間と同じくらい、リクエスト当たりの文の数も注意して見ます。

最後の一個を、二度売ります。 二つのリクエストが、同じ在庫数を読み、どちらも十分あると判断して書き込みます。読み取りと書き込みを別々の文にすると、確認と実行の間に隙間が生まれ、同時に来る通信が、その隙間へ入り込みます。先ほどの条件付きUPDATEとRETURNINGを使い、クエリーの中で隙間を閉じます。あるいは、SELECT … FOR UPDATEで取得したロックの保持時間をできる限り短くし、ロック取得からコミットまでの間に、ネットワーク呼び出しを挟まない方法を取ります。

調査で、データを変えてしまいます。 遅いDELETEの計画を見るためにEXPLAIN ANALYZEを実行したら、行が消えます。ANALYZEは、実際に文を実行し、出力だけを捨てるため、すべての副作用が起きます。データを変更する文を解析するときは、BEGINとROLLBACKで囲みます。あるいは、見積もりで十分なら、実行せずに計画だけを作る、通常のEXPLAINを使います。

参考資料

- SELECT — 各句の処理順を含む、PostgreSQLのクエリーの完全な構文です。
- 7.1. 概要 — クエリーについての章の冒頭で、用語を定義しています。
- 51.1. クエリーの経路 — クライアントが送った文が、パーサー、リライター、プランナーを通り、エグゼキューターへ届くまでです。
- 14.1. EXPLAINの使い方 — 計画の読み方、コストの単位の意味、ANALYZEが文を実行する理由です。
- F.32. pg_stat_statements — SQLの計画と実行の統計を追跡する — 合計コストでクエリーを順位付けする拡張です。
- 13.2. トランザクションの分離 — READ COMMITTEDで二人目の書き手に何が見え、なぜWHERE句を再評価するのかを説明します。

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

「クエリーとは？」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/query/