PostgreSQL インサイド

チューニング

SQL チューニングの概要

技術を知る

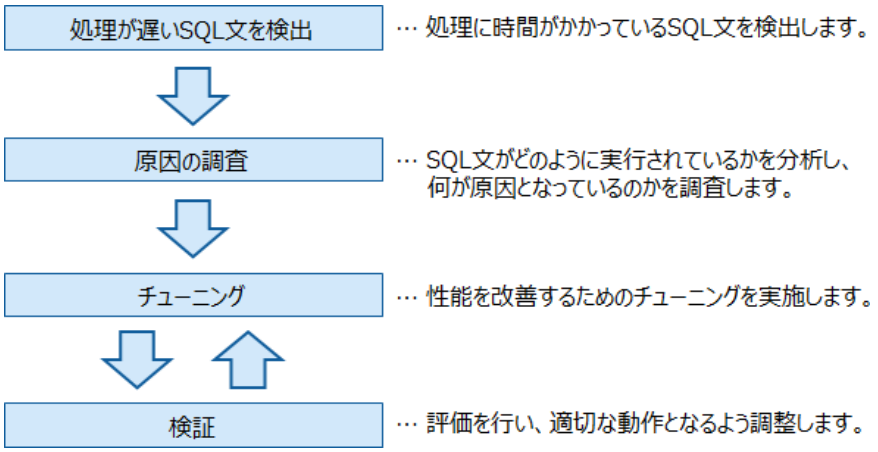
FUJITSU

- [ ] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [x] チューニング
- [ ] バックアップ／リカバリー
- [ ] 冗長化／負荷分散
- [ ] 監視
- [ ] データ連携
- [ ] 災害対策
- [ ] 豆知識

SQL チューニングとは、特定の SQL などを対象に局所的に調整をし、解決していくことです。ここでは、PostgreSQL における「SQL チューニングの概要」と、「処理が遅い SQL 文の検出」「原因の調査」について解説します。

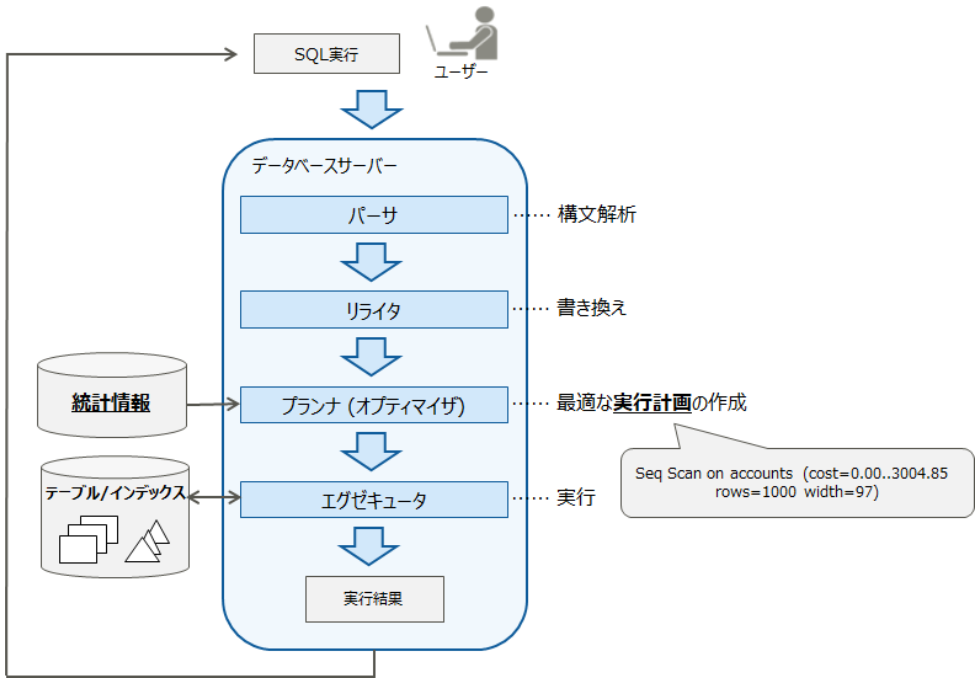
1. SQL チューニングとは

SQL チューニングは、SQL 実行で性能に問題が発生した場合に、SQL の内部処理を解析して、最適な動作となるように改善することを目的としています。処理が遅い SQL 文を対象としているため、開発したアプリケーションを検証しながら行います。チューニングと検証を繰り返し、適切な動作となるようにしていきます。
以下に、SQL チューニングの流れを示します。



2. SQL が実行される仕組み

SQL のチューニングについて解説する前に、まず、データベースサーバーで SQL が実行される仕組みについて理解しておく必要があります。ここでは、SQL 文が実行されてから結果が返ってくるまでの流れについて解説します。
クライアントから SQL が実行されると、その文字列が正しい構文になっているかチェックします。それが「パーサ」です。次に、データベースにルール（SQL を置き換える規則）が定義されている場合、「リライタ」がその規則に従って、他の SQL に書き換えます。この解析・書き換えられた SQL を、「プランナ（オプティマイザ）」が、どのように実行するか、『統計情報』を参照しながら決めます。統計情報には、どのテーブルにどのくらいデータが入っているという要約情報が入っています。この情報と、postgresql.conf の情報を参照して、最適な SQL の『実行計画』を作成します。作成された実行計画を、「エグゼキュータ」が実行し、結果を返します。



SQL を実行してから結果が返るまでの処理時間が短くなる決め手は、「統計情報から良い実行計画が作成されているか」ということになります。この統計情報と実行計画について、もう少し詳しく説明します。

2.1 統計情報とは

PostgreSQL は、データベース内のテーブルやインデックスの行数やサイズ、各列のデータの重複度合いや出現頻度、分布状況などを利用して、SQL の最適な実行手順（方法）を決定します。最適な実行手順（方法）を決定するための情報のことを「統計情報」といい、PostgreSQL が収集・記録しています。
統計情報の収集および最新化は、自動バキューム処理の実行時に自動的に行われます。PostgreSQL 8.3 以降では、デフォルトで自動バキューム処理が実行される設定（postgresql.conf の autovacuum パラメーターが有効）になっています。
データベースを運用してデータが更新されていくに従って、統計情報の内容は古くなっていきます。古い統計情報を基にして SQL の最適な実行手順（方法）を決定しても、現在のデータにアクセスするためには不適切で非効率になる可能性があります。また、バッチ処理などでデータを大量にロードした場合や一括更新した場合は、統計情報と実データに大きな差が発生するので、自動バキューム処理が行われるまで待つのではなく、ANALYZE コマンド（SQL コマンド）を実行することで、統計情報を最新化してください。

2.2 実行計画とは

PostgreSQL は、統計情報を基に、クライアントから実行された SQL をどのようにしたら短い時間で実行できるか、いくつかの検索方法、結合方法、および、結合順序の組み合わせ候補から、最も早くてコストの低い方法を選択します。この選択された方法のことを「実行計画」といいます。
PostgreSQL が作成した実行計画は、必ずしも最適な実行計画とは限りません。先に述べた通り、統計情報が古い場合もありますが、アプリケーションなどの予期しない負荷や挙動、各種リソース不足が原因になる場合もあります。そのため、SQL のチューニングは、データベースを構築する上で、とても重要な作業と言えます。
PostgreSQL が作成した実行計画に問題がないかを確認する方法については、「4. 原因の調査」で説明します。

3. 処理が遅い SQL 文の検出方法

処理が遅い SQL 文を検出する方法について説明します。
処理が遅い SQL 文を検出する方法には、以下の 2 つがあります。これらを利用するためには、事前の設定が必要です。ここでは、事前の設定方法について説明します。

- 統計情報ビューを利用して検出する方法
- サーバーログを利用して検出する方法

3.1 統計情報ビューを利用して検出する方法

pg_stat_statements は、サーバーで実行されたすべての SQL 文の実行時の統計情報を記録します。この情報は、pg_stat_statements ビューにより、SQL 文を実行したユーザーの OID（userid）、実行された SQL 文（query）、SQL 文の実行回数（calls）、SQL 文の実行に費やした総時間（total_time（注 1））などを確認することができます。
pg_stat_statements ビューを使用するには、追加モジュールを公開サイトなどから取得してインストールする必要があります。
追加モジュールをインストールするために、以下を実行してください。

- 注 1　PostgreSQL 13 から、パラメーター名が total_exec_time に変更されました。

```
CREATE EXTENSION pg_stat_statements;
```

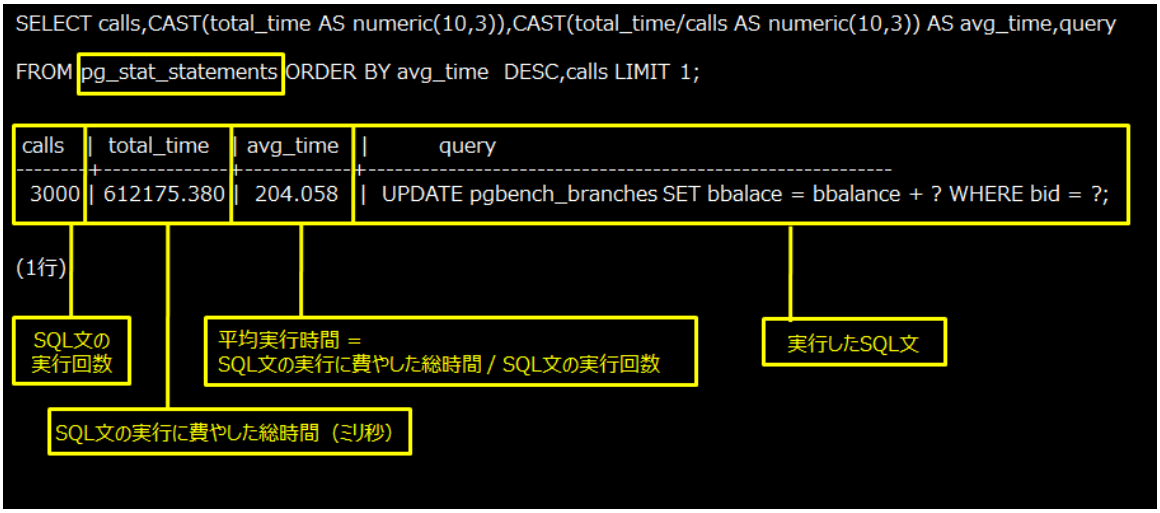
ポイント

Fujitsu Enterprise Postgres では、製品にプリインストールしているので、追加モジュールを取得する必要はありません。

また、postgresql.conf の shared_preload_libraries パラメーターに pg_stat_statements を追加し、関連する以下のパラメーターを設定しておく必要があります。設定後、サーバーを再起動することで、pg_stat_statements ビューが使用できるようになります。

shared_preload_libraries	pg_stat_statements を指定
pg_stat_statements.max	情報を保持する SQL 文の最大数
pg_stat_statements.track	以下のいずれかを指定 • top：クライアントが直接実行した SQL 文のみ記録 • all：関数内から呼び出されたものも含めてすべて記録 • none：記録しない
pg_stat_statements.track_utility	以下のいずれかを指定 • on：SELECT、INSERT、UPDATE、DELETE 以外のユーティリティーコマンドも記録 • off：SELECT、INSERT、UPDATE、DELETE のみ記録
pg_stat_statements.save	以下のいずれかを指定 • on：サーバー終了時に統計情報を保持 • off：サーバー終了時に統計情報を保持しない

【例】 pg_stat_statements ビューで、平均実行時間が最も長い SQL を調べた場合は、以下のようになります。



pg_stat_statements により SQL 文に関する情報を記録し続けるとパフォーマンスが低下するため、チューニングが終わったら、「pg_stat_statements.track=none」（記録しない）を設定し、サーバーを再起動してください。

3.2 サーバーログを利用して検出する方法

SQL 文の実行が設定した時間以上かかった場合、その SQL 文と実行に要した時間を、サーバーログにメッセージとして出力させることができます。デフォルトでは遅い SQL 文の情報はサーバーログには出力されません。情報を出力させるには、postgresql.conf に log_min_duration_statement パラメーターを設定します。このパラメーターに設定した値以上の時間がかかった SQL 文について、実行時間とその SQL 文が、メッセージとして出力されます。

log_min_duration_statement	以下のいずれかを指定 • 0 以上：設定した値以上の時間がかかった SQL 文と実行時間を出力（0 の場合は、すべての SQL 文と実行時間が出力） • -1：SQL 文と実行時間を出力しない

例えば、実行に 3 秒以上かかった SQL を出力させるには、「log_min_duration_statement= 3s」のように設定します。
短い時間を設定すると、多くのメッセージが出力されることになります。大量のメッセージがサーバーログに出力されると、システム全体の性能が劣化することがあるため設定する値には注意してください。

【例】

サーバーログには以下のように出力されます。

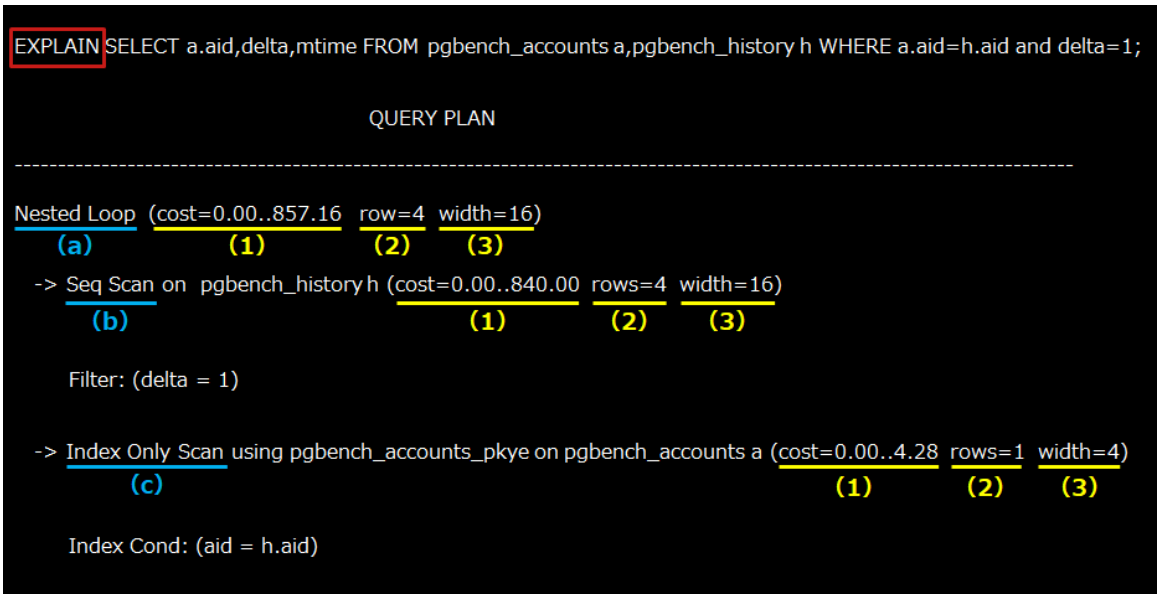
また、追加モジュールの auto_explain を用いることで、設定した値以上の時間がかかった SQL 文の実行計画もサーバーログに出力させることができます。auto_explain の詳細については、「PostgreSQL 文書」の付録を参照してください。

4. 原因の調査

処理が遅い SQL 文を検出できたので、次はその原因を調査します。
検出された SQL 文の先頭に「EXPLAIN コマンド」を付けて実行することで、その SQL の実行計画を表示することができます。この結果をもとに原因の調査を行います。

4.1 EXPLAIN コマンドによる実行計画の表示

「EXPLAIN コマンド」により、SQL 文が参照するテーブルをスキャンする方法や、取り出した行を結合する方法が表示されます。「EXPLAIN」だけを付けて実行した場合は、その SQL 文を実行するのにかかるコスト、行数、入力サイズの見積りが表示されます。実際に SQL 文は実行されないため、推測の値が表示されます。



（a）、（b）、（c）

SQL 文が参照するテーブルをスキャンする方法や、取り出した行を結合する方法が表示されます。

- （a）pgbench_history テーブルと pgbench_accounts テーブルを、Nested Loop と呼ばれる方式（二重ループによるテーブル結合）で結合していることがわかります。
- （b）pgbench_history テーブルは、「Seq Scan」と呼ばれる方式（テーブルのすべての行を上から順に調べる方式）を選択していることがわかります。
- （c）pgbench_accounts テーブルは、「Index Only Scan」と呼ばれる方式（インデックスのみにアクセスし、テーブルにはアクセスしない方式）を選択していることがわかります。

（1）、（2）、（3）

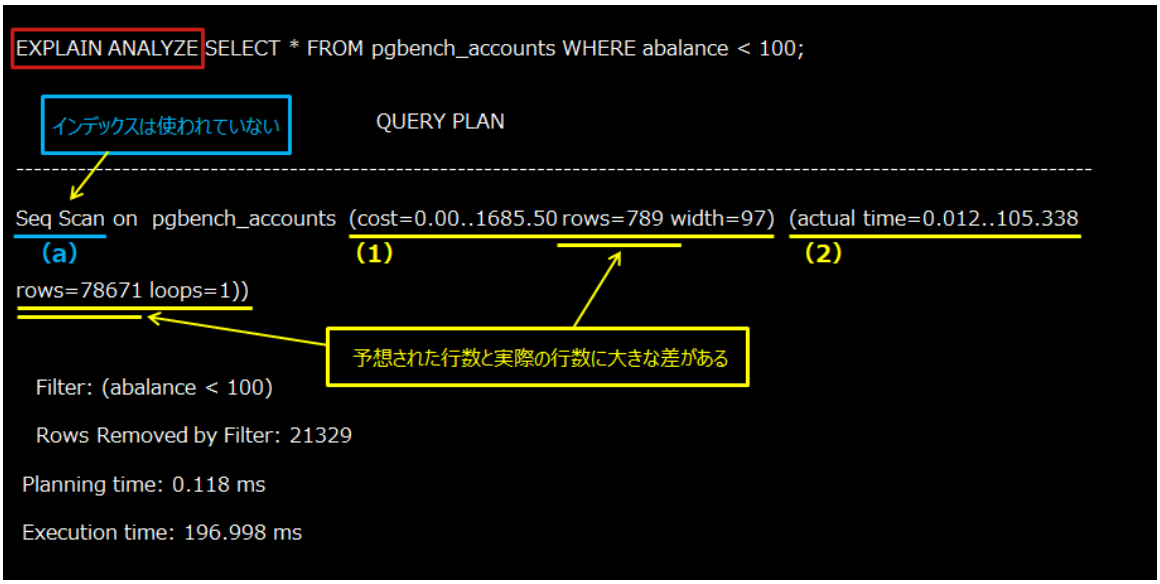
その SQL 文を実行するのにかかるコスト、行数、入力サイズの見積り（推測の値）が表示されます。

- （1）最初の行を返すまでのコスト（左側）と、すべての行が返し終わるまでのコスト（右側）がわかります。この値には、通信にかかるコストやクライアント端末の表示コストは含まれていません。
- （2）推測された問い合わせ結果の行数がわかります。
- （3）推測された入力サイズがわかります。

4.2 ANALYZE オプションによる実測の値の表示と確認方法

「EXPLAIN コマンド」に「ANALYZE オプション」を付けることで、推測の値に加え、実測の値も表示されます。SQL 文を実行した値を表示することで、推測の値と、実測の値が近いかどうかを確認することができます。
実際に SQL 文が実行されるため、INSERT 文や DELETE 文などを行う際は、データに影響を与えないよう、以下のように指定してください。

```
BEGIN;
EXPLAIN ANALYZE ...;
ROLLBACK;
```



（a）

SQL 文が参照するテーブルをスキャンする方法や、取り出した行を結合する方法が表示されます。
pgbench_accounts テーブルは、「Seq Scan」と呼ばれる方式（テーブルのすべての行を上から順に調べる方式）を選択していることがわかります。「Seq Scan」と表示されているので、インデックスは利用されず、テーブルのすべての行を調べることを意味します。行数が多いテーブルに対しては、インデックスを付加することで、性能向上が見込まれます。インデックスを利用した場合は、「Index Scan」と表示されます。

（1）、（2）

その SQL 文を実行した際の推測の値と実測の値が表示されます。
PostgreSQL は、統計情報の値を基に処理行数を推測します。（1）の推測の値の rows=789 が、（2）の実測の値の rows=78671 と大きく異なるということは、統計情報が古くなっていることを示しています。手動で、統計情報の最新化（ANALYZE コマンドを実行）を行うことを推奨します。

SQL チューニングの概要と、処理が遅い SQL 文の検出方法および原因の調査について解説しました。処理が遅い SQL 文を見つけ、チューニングにより処理を改善してください。

2021 年 1 月 22 日