

# SQL チューニングアプローチの考え方

## 技術を知る

- |                                   |                             |                                |  |   |
|-----------------------------------|-----------------------------|--------------------------------|--|---|
| <input type="checkbox"/> 導入／環境設定  | <input type="checkbox"/> 移行 | <input type="checkbox"/> 性能    | <input checked="" type="checkbox"/> チューニング | <input type="checkbox"/> バックアップ／リカバリー   |
| <input type="checkbox"/> 冗長化／負荷分散 | <input type="checkbox"/> 監視 | <input type="checkbox"/> データ連携 | <input type="checkbox"/> 災害対策              | <input checked="" type="checkbox"/> 豆知識 |

SQL チューニングアプローチの考え方として、テーブル定義や SQL 文の書き方を見直す考え方と、PostgreSQL のプランナ（オプティマイザ）が作成する実行計画を外部から制御する考え方があります。後者には、SQL 文にヒント句を指定してアクセス方法を明示的に指示する手法と、統計情報を固定化して常にその統計情報が利用されるようにする手法があります。ここでは、SQL チューニングアプローチ手法の概要と、実行計画を外部から制御する手法の使い分けについて解説します。

### 参考

SQL チューニングの概要や用語については「SQL チューニングの概要」で詳しく解説しています。

### SQL チューニングアプローチ手法の紹介

SQL チューニングには、具体的に以下 3 つのアプローチ手法があります。

#### 1) テーブル定義と SQL 文の書き方を見直す

テーブルやインデックスの定義、WHERE 句や条件の記載を適切に見直すことにより、PostgreSQL のプランナ（オプティマイザ）が最適な実行計画を作成するように改善します。なお、プランナ（オプティマイザ）は現在の統計情報を元に実行計画を作成するため、運用においては、統計情報が最新化されている必要があります。

#### 2) 周辺 OSS の pg\_hint\_plan でヒント句を使って実行計画を制御する

複雑な SQL を実行する場合や、テーブルの更新頻度が高く統計情報を最新化できていない場合、最適な実行計画が作成されないことがあります。その場合は、ユーザーが SQL 文にヒント句を追加することにより、意図した実行計画になるよう制御します。

#### 3) 周辺 OSS の pg\_dbms\_stats で統計情報を固定化して実行計画を制御する

刻々と統計情報が変化する場合、最適な実行計画が作成されないことがあります。その場合は、ユーザーが統計情報を固定化することにより、間接的に実行計画を制御します。

SQL チューニングが必要になった場合、まず、1) の手法で SQL チューニングを実施することが重要です。それでも解決できない場合には、周辺 OSS を利用した 2) や 3) の手法を実施することを検討します。

### 参考

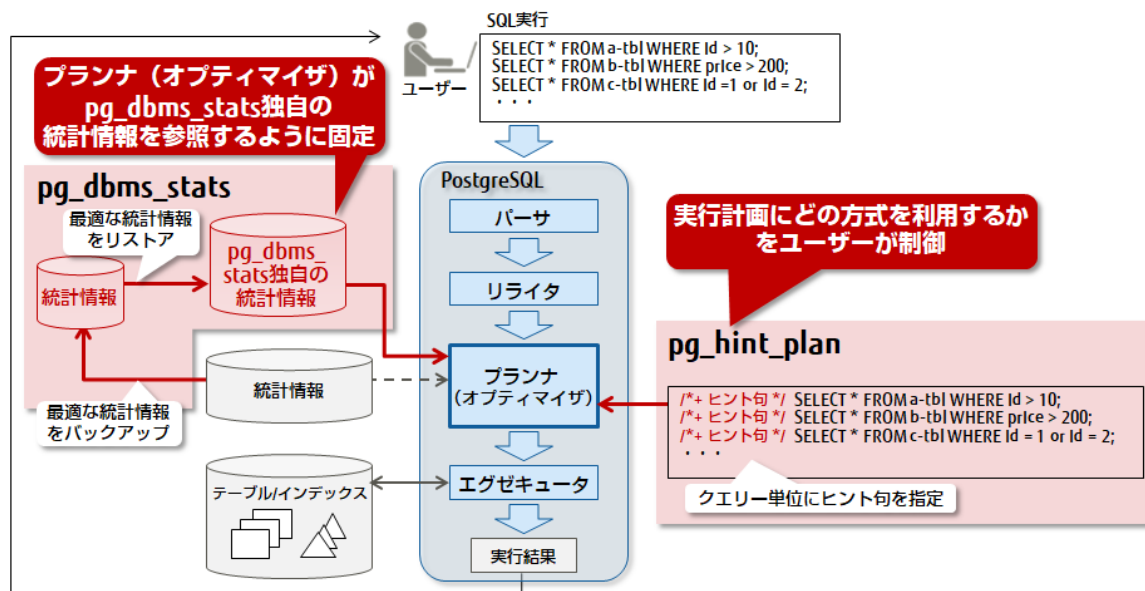
各手法の操作例については、PostgreSQL 技術インデックスで詳しく解説しています。

- 1) のテーブル定義を見直す操作例については、「SQL チューニングを実施する」
- 2) の操作例については、「pg\_hint\_plan で実行計画を制御する」
- 3) の操作例については、「pg\_dbms\_stats で統計情報を固定化して、実行計画を制御する」

2) と 3) の手法は、どちらもプランナ（オプティマイザ）の作成する実行計画を外部から制御する周辺 OSS を使用しますが、使い分けの判断が難しいところがあります。次に、これら 2 つの手法の仕組みと使い分けについて説明します。

### pg\_hint\_plan と pg\_dbms\_stats の仕組み

ユーザーが入力した SQL 文が実行される仕組みと、その過程で pg\_hint\_plan と pg\_dbms\_stats が作動する仕組みについて、以下に図解します。



### pg\_hint\_plan と pg\_dbms\_stats の使い分け

pg\_hint\_plan と pg\_dbms\_stats が、それぞれがどのようなケースに適しているかについて解説します。一般的には、業務アプリケーションが改修できるか否か、チューニングの単位を各クエリとするかクエリ全体とするかで使い分けられます。

#### pg\_hint\_plan の使いどころ

pg\_hint\_plan は、一般的に業務アプリケーションの改修量が少ない、影響箇所が局所的である、改修の工数が確保できるなど、個々に SQL 文を改修できる場合に利用されます。クエリ単位にヒント句を指定することで、SQL 文や GUC パラメーターを変更せずに、実行計画を制御できます。ヒント句には、スキャン方法や結合順などを記載しますが、SQL 文の実行結果には影響しません。そのため、テストはヒント句を付けた SQL 文の性能テストだけで済みます。

#### pg\_dbms\_stats の使いどころ

pg\_dbms\_stats は、業務アプリケーションの数が大量である、影響箇所が広範囲に及ぶ、改修の工数が十分に確保できないなど、個々に SQL 文を改修できない場合に利用されます。大規模で性能の安定化が求められるシステムなどで利用実績があります。性能テストは統計情報を固定化している際に実行されるすべての SQL 文を対象に行う必要があります。

2020 年 6 月 12 日