Feita para Desenvolvedores de Software e DBAs

SQL magazine

Edição 139 :: R$ 14,90

DEVMEDIA

ISSN 1677918-5

Desenvolvimento
Configuração de bases de desenvolvimento, testes e homologação

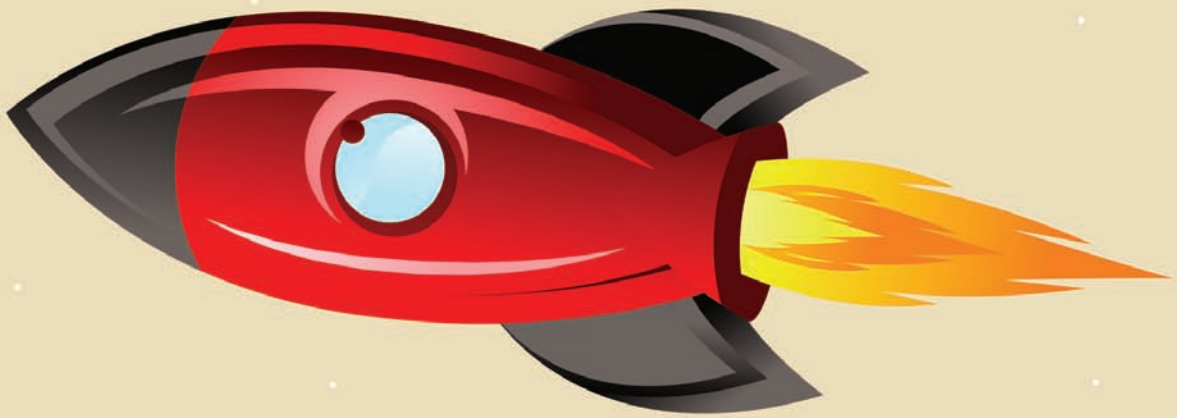
SQL Server 2016
Conheça o Stretch Database

MySQL
Trabalhando com múltiplas instâncias

# OTIMIZAÇÃO DE DESEMPENHO

Tuning lógico do SQL Server

Consultas otimizadas no PostgreSQL



Banco de dados
Acessibilidade das ferramentas de bancos de dados

Oracle
Torne seu banco mais robusto com o Archivelog Mode

Oracle
Configurando uma replicação de dados bidirecional heterogênea

# Queries saudáveis no PostgreSQL

## Técnicas e métodos para otimizar consultas


### Fique por dentro

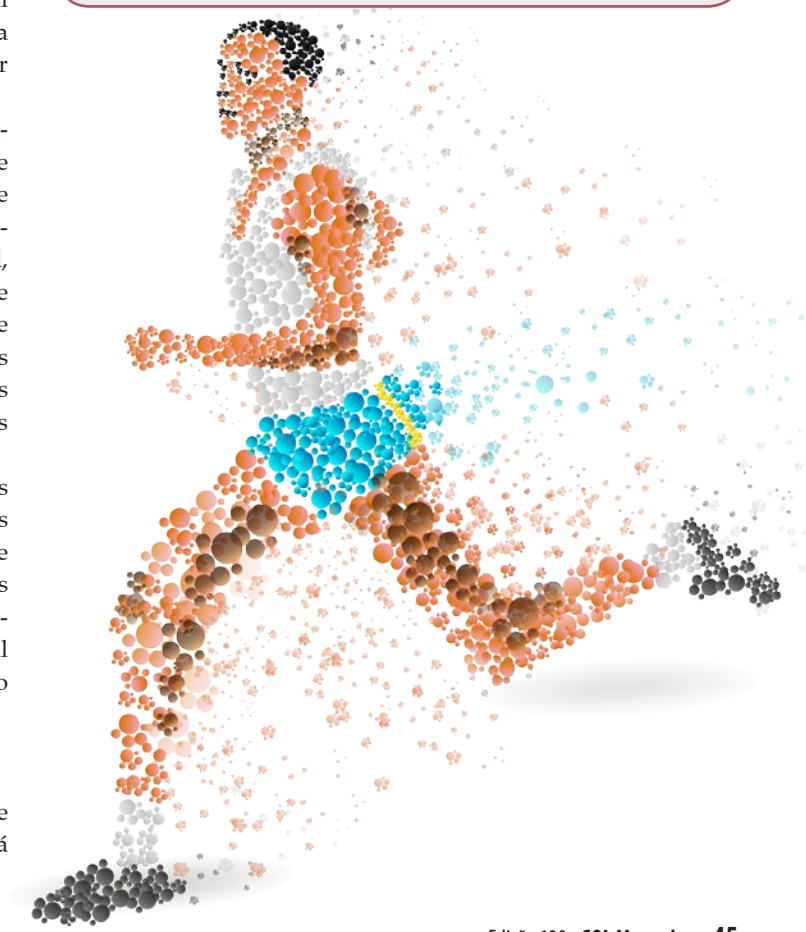
Este artigo apresenta técnicas e boas práticas para a construção de queries no banco de dados PostgreSQL, além de apresentar como essas consultas podem ser cada vez mais rápidas sem onerar a integridade e disponibilidade do seu banco de dados. Este artigo será útil para situações nas quais se deseja obter uma performance ótima na execução de consultas. Existem alguns métodos no PostgreSQL que, se utilizados de maneira correta, facilitam o melhor desempenho das queries. Por exemplo, o EXPLAIN auxilia no entendimento dos relacionamentos e indexações das estruturas do banco de dados. Além disso, para melhorar o desempenho, algumas boas práticas bem empregadas podem ajudar no objetivo de construir uma query eficiente.


Hoje em dia, qualquer sistema que seja utilizado em diferentes áreas do mercado possui um gerenciador embarcado que administra um repositório de informações em uma camada não aparente para o usuário final. Esses sistemas são chamados de bancos de dados e possuem caráter extremamente relevante e valioso para quem os utiliza. Com o avanço empregado nessa forma de armazenamento de informações, as mesmas se tornaram cada vez mais expansíveis e robustas para se adaptarem à grande demanda de consumo nas estruturas existentes.

O principal destaque deve ser dado à manipulação que será realizada nesses sistemas. Elencar e identificar quais são as reais necessidades é um fator primordial para a qualidade do produto desse processo. Ou seja, a instrução SQL construída deve ter utilidade e atender às necessidades do seu cliente.

O desejo de qualquer um dos usuários, seja ele desenvolvedor ou administrador, que utilizam um banco de dados, sendo ele PostgreSQL ou qualquer outro tipo de SGDB (SQL ou até do tipo NoSQL), é que o tempo de resposta de suas consultas SQL seja o mais rápido possível, quase sempre com retornos imediatos independente de estarmos lidando com pequenas porções de dados, que possuem um número mínimo de tabelas, ou com bases de dados gigantescas, altamente estruturadas, repletas de vínculos internos e externos e com muitas interações simultâneas executadas diariamente.

É importante destacar que todos os métodos e técnicas de otimizações demonstradas neste artigo são sugestões para um aumento no desempenho de queries dentro de uma estrutura de banco de dados PostgreSQL. Todas elas, quando bem aplicadas no contexto onde se encaixam e com a necessidade empregada no processo no qual pertencem, surtirão um resultado satisfatório quando o assunto for o quão veloz é sua consulta SQL.



### Fatores primordiais

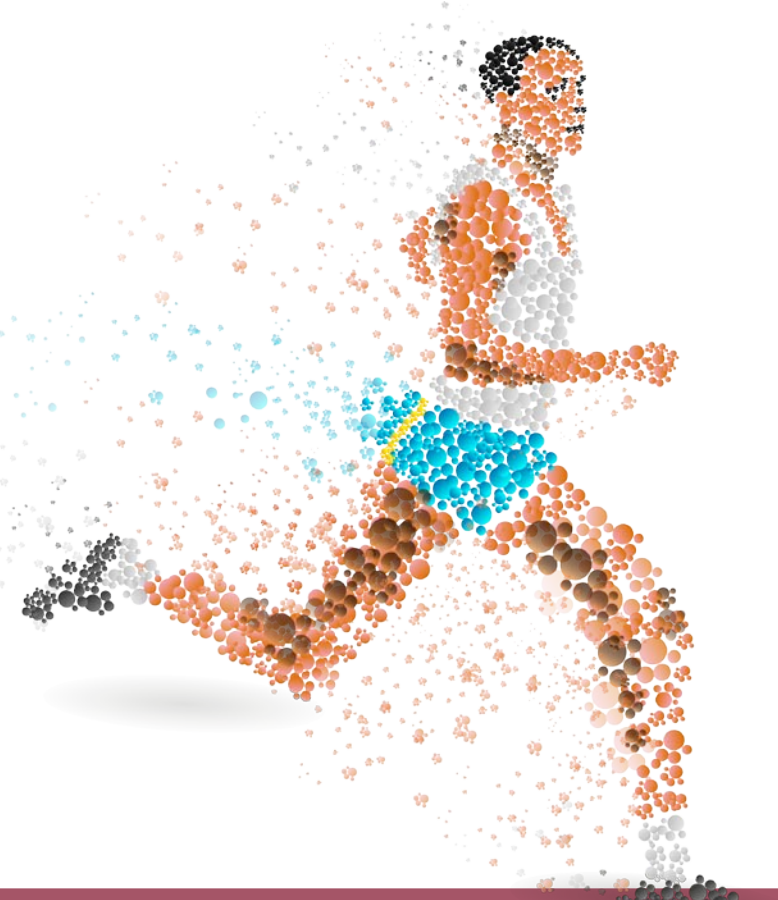
Quando se fala em otimização de queries, sempre se deve levar em consideração o ambiente no qual se está

trabalhando. No melhor dos cenários, quando evidenciamos o aprimoramento de instruções SQL, alguns fatores são primordiais para que os métodos empregados para aumentar a velocidade de resposta das consultas ajam de maneira correta dentro do servidor e que surtam os resultados esperados. Dentre eles, podem-se evidenciar dois tópicos que estão presentes em todos os cenários onde uma instrução SQL não traz os resultados dentro do tempo que se espera.

O primeiro deles está na projeção e na forma na qual foi idealizado o banco de dados onde se está trabalhando. É possível afirmar que esse sempre será o principal gargalo quando o desempenho das consultas executadas nessa base de dados não for eficiente. A formação das estruturas de um banco de dados terá um grande reflexo no comportamento de escrita e leitura dos registros no disco onde a base está alocada, ou seja, tabelas que sejam mal estruturadas podem prejudicar seriamente o desempenho das queries.

O segundo fator, mas não de menor importância, está na presença de uma estruturação de índices adequada. Índices podem reduzir drasticamente o tempo de acesso a uma tabela do banco de dados e também diminuir a intensidade de acesso aos discos físicos. Isso ocorre devido aos padrões de acesso criados por eles nas páginas de dados referentes às tabelas que são utilizadas no banco; um caminho mais curto comparado com a varredura de um arquivo inteiro.

Para cada instrução SQL que é executada, temos no contexto do servidor um novo processo disparado e uma nova carga de dados que será posta em cheque para as diversas relações e comparações. Quando o resultado for alcançado, serão apresentados blocos pré-selecionados, que podem ser chamados de relatórios e que servirão para uma posterior análise, tomada de decisão e readequação estratégica. O alto desempenho dos processos de execução do banco sempre caminha lado a lado com a qualidade do produto.



## Processo de otimização

Veremos agora alguns métodos que podem ser empregados no processo de otimização de instruções baseadas no PostgreSQL. Com esses recursos, será possível criar consultas mais inteligentes com uma otimização e alta performance.

Toda otimização deve ser bem contextualizada, ou seja, deve levar em consideração o ambiente no qual os dados necessários estão sendo manipulados, bem como a consciência do contexto da técnica que está sendo aplicada. O processo de aprimoramento propõe uma sistemática que pode ser demorada e exigirá uma experiência de campo do usuário que estiver trabalhando na construção da query. Quanto mais esforço e tempo empregarmos nas otimizações, melhores serão os resultados encontrados na conclusão do processo.

Apresentaremos na sequência algumas ferramentas pertencentes ao PostgreSQL que, a partir da intensidade de leitura e escrita, dos conceitos de padrões de estruturação e da manutenção das indexações existentes, podem ajudar no trabalho de otimização de queries.

## Mecanismo de planejamento

A partir do momento que uma instrução SQL é executada em uma ferramenta de interface do PostgreSQL, por exemplo, o servidor realizará, internamente, uma chamada ao sistema gerenciador do banco de dados que, por sua vez, desempenha uma preparação prévia para saber quais decisões serão tomadas em relação às tabelas que necessitarão de leitura e/ou escrita, de qual forma serão tomadas e como ocorrerão os seus relacionamentos. Abrangendo todos os tipos de queries (select, insert, update etc.), esse mecanismo interno implementa uma fatoração de todos os elementos, juntamente de seus atributos utilizados para a construção da query, com o objetivo de sumarizar o acesso das tabelas de uma maneira mais elaborada. Essa idealização é chamada de Plano de Execução (ou Plano de Consulta, ou *Explain Plan*).

O plano de execução pode ser visto como uma sequência de passos que o SGBD precisará percorrer para finalizar a execução da instrução SQL, ou seja, desvendar quais tabelas poderão ter seus índices utilizados ou se essas estruturas sofrerão uma varredura linear e completa, além de realizar a estatística de todo o procedimento, informando o tempo de entrada e tempo de saída de cada laço da execução, a quantidade de registros envolvidos na transação e de blocos de disco físico afetados por cada tabela.

## Consumindo o plano de execução

Vista friamente, uma consulta que envolve a listagem de todas as colunas de uma tabela do banco de dados, sem nenhum filtro que remova registros pertencentes a essa estrutura, é considerada uma instrução relativamente simples para um usuário. Contudo, o SGBD realiza internamente um trabalho oneroso, que pode envolver diversos arquivos e páginas de dados dentro do servidor a cada comando recebido através da instrução SQL.

A demonstração do plano de execução dentro do PostgreSQL pode ser vista através do comando EXPLAIN. Essa instrução apresenta na tela todo o plano de execução criado pelo planejador do SGBD para a query fornecida no console, quais algoritmos de junção foram utilizados, caso haja relacionamento entre tabelas, se há a necessidade explícita de ordenação de colunas e como foi realizada a varredura pelas tabelas referenciadas. A **Figura 1** demonstra um exemplo do comando EXPLAIN e seus resultados.

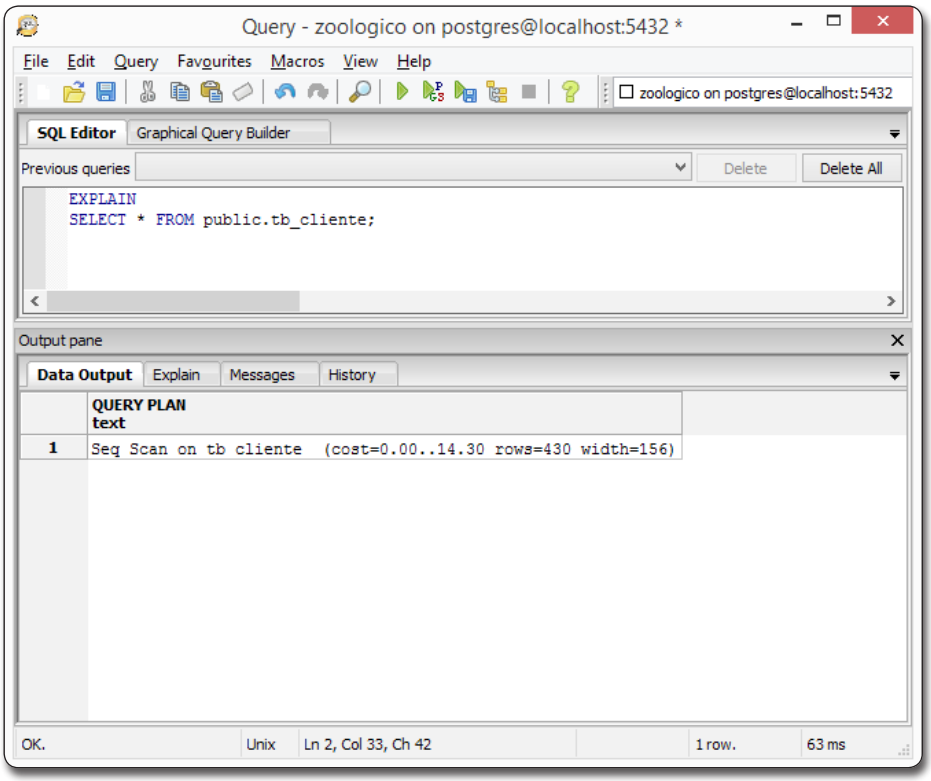



**Figura 1**. Exemplo de utilização do comando EXPLAIN

Podemos interpretar os resultados do EXPLAIN apresentado na **Figura 1** da seguinte forma:

- Seq Scan on tb_cliente: mostra que foi realizada uma pesquisa sequencial simples pela tabela 'tb_cliente';
- Cost=0.00..14.30: mostra o custo inicial e custo final estimado para a execução do comando;
- Rows=430: evidencia a quantidade de linhas que o planejador espera encontrar no resultado da query;
- Width=156: evidencia o tamanho médio que o planejador estima das linhas que retornarem da query (em bytes).

O custo apresentado pelo comando EXPLAIN é uma estimativa realizada pelo planejador que indica quanto tempo será gasto na execução da instrução SQL. Para chegar nesse valor, é verificada inicialmente a quantidade de páginas de disco acessadas para somar a quantidade de linhas percorridas. Esse indicador varia conforme a mudança dos fatores estipulados, chamados de constante de custo e que também podem mudar dependendo do tipo de varredura. Na maioria dos casos, o tempo total será o mais importante para a análise do resultado.

Para o conhecimento do número de páginas em que uma tabela está disseminada (ver **Figura 2**), é possível acessar os dados da tabela pg_class, estrutura que faz parte das tabelas de sistema do PostgreSQL. Além desse número, a tabela apresenta informações de todas as estruturas existentes no banco, índices e visões.

Para obter resultados precisos nas execuções das instruções SQL, geralmente são utilizados comandos para restringir registros que não são necessários no resultado final da query. Geralmente, essa técnica é realizada através do comando WHERE com o intuito de filtrar linhas. Em relação ao desempenho, deve-se ter cuidado ao fazer o uso desse tipo de comando.

Ao contrário do que é imaginado por muitos usuários, fazer uso desse tipo de instrução corriqueiramente não diminui o tempo de execução de uma consulta, mas sim, causa o aumento do tempo gasto para processar a instrução. O motivo para esse aumento está na verificação pertencente à cláusula WHERE somada ao tempo gasto para a leitura sequencial realizada na tabela em questão. A **Figura 3** apresenta o resultado do plano de execução de uma consulta com a cláusula WHERE especificada. Observe que o plano é diferente do apresentado para a consulta sem o uso do WHERE na **Figura 1**.

É possível observar que, apesar de reduzir o número estimado de linhas do resultado e manter a largura média (width), o novo custo de execução estimado pelo planejador sofreu um aumento se comparado ao número apresentado na consulta sem as cláusulas de restrição. Esse tipo de alteração acontece porque o SGBD necessita percorrer todas as linhas pertencentes à tabela, além de atender à condição especificada no comando WHERE que, por ventura, deixa a instrução mais custosa para o servidor, refletindo também no tempo de processamento de CPU. Comparados os tempos, a diferença apresentada pode ser expressivamente maior caso a quantidade de linhas retornadas e filtradas aumentem.

## Utilize bem os índices

Os bancos utilizam blocos de disco para realizar o armazenamento dos registros dentro do servidor, blocos esses que possuem uma organização através de páginas de dados. A cada instrução SQL executada, todas as páginas que guardam os dados das tabelas referenciadas na consulta são varridas. São nessas páginas onde se encontram todas as linhas que o PostgreSQL realizará a pesquisa para fornecer o resultado. Existem dois tipos de varredura de páginas: uma com o método sequencial, na qual a varredura acontece em todas as linhas, e outra com a utilização de índices, na qual a busca pelos registros é melhor direcionada e não há a necessidade de fazer a verificação de todas as linhas.

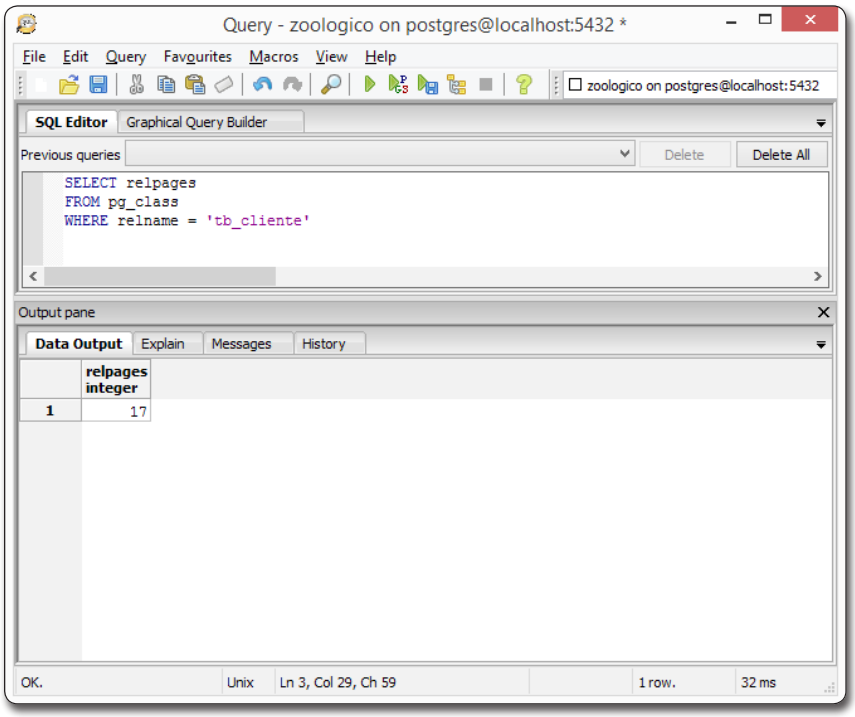



**Figura 2**. Consulta do número de páginas na tabela pg_class

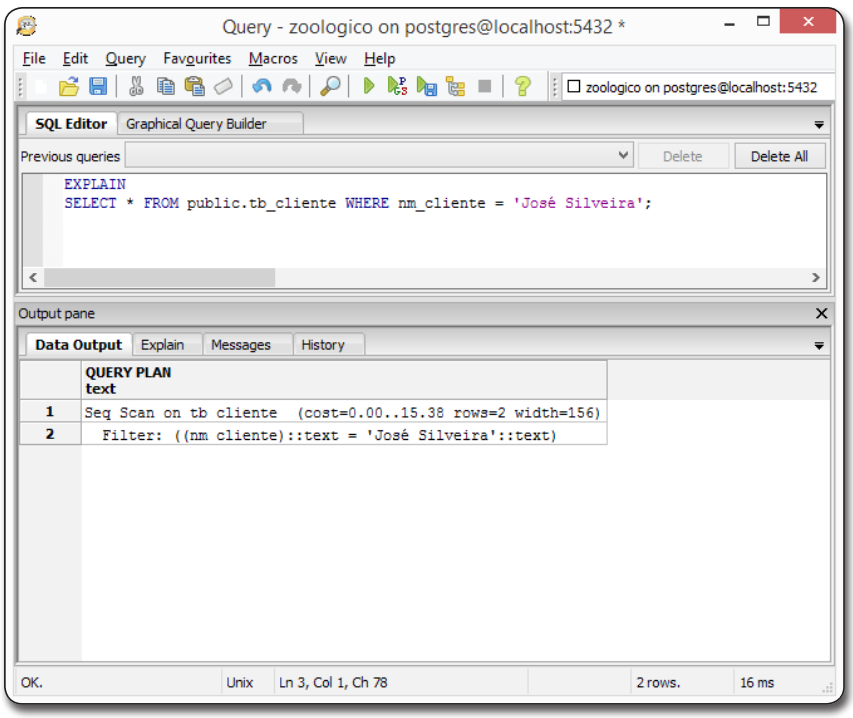



**Figura 3**. Exemplo de plano de execução com a cláusula WHERE

Apesar de sua famosa comodidade e versatilidade, os índices só podem ser utilizados quando o resultado que é retornado de uma instrução não ultrapassar 5% do número total de registros armazenados na tabela. Isso porque índices são aconselhados para tabelas com um bom nível de seletividade. Caso contrário, pode até se tornar menos custoso fazer uma varredura total da tabela. Durante a criação do plano de execução, o planejador realiza essa análise internamente para verificar qual dos métodos de varredura terá o melhor ganho de custo estimado no momento da execução da query.

Para exemplificar melhor esses conceitos, será apresentado o resultado de um plano de execução realizado sobre uma consulta SQL, onde existem duas condições bem específicas na cláusula WHERE. A **Figura 4** mostra como o planejador estima o resultado dessa instrução.

Como é possível analisar no último exemplo, apesar de existirem duas condições evidenciadas na cláusula WHERE, o planejador do SGBD decide utilizar a varredura através dos índices existentes sobre os campos, pois esse método de leitura gera um menor custo estimado de execução em relação ao método sequencial. Ou seja, como a primeira condição estabelecida na cláusula WHERE possuía um índice criado sobre a coluna, ela foi selecionada como condição do índice, e o filtro da seleção dos registros retornados foi realizado apenas sobre a segunda condição estabelecida.

Em um cenário ideal, é importante analisar na consulta se todos os principais campos que podem ocorrer alta seletividade possuem índices criados. Em uma listagem ordenada, é importante garantir a indexação nestes seguintes atributos:

- Chaves primárias;
- Chaves estrangeiras;
- Campos filtrados, onde ocorram condições da cláusula WHERE, por exemplo;

• Campos agregados, onde haja algum tipo de condição de agregação (SUM, AVG etc.);
• Campos ordenados, onde ocorram evidencias na cláusula ORDER BY, por exemplo.

É importante salientar que o fato de uma instrução SQL possuir alta seletividade, ou seja, realizar a utilização dos índices quando eles houverem, não garante que o tempo e custo da execução seja menor do que o praticado no método sequencial. Casos onde não é levado em consideração o alto número de linhas retornadas, o resultado pode ser mais oneroso do que uma varredura sem o uso de índices.

## Entendendo o comando EXPLAIN

Existem diferentes opções de utilização do comando EXPLAIN dentro do PostgreSQL, cada uma com uma característica específica. Uma das variações mais utilizadas, o comando EXPLAIN ANALYZE, possui uma diferença considerável em relação ao seu comando base.

A utilização da variação ANALYZE possibilita uma análise mais precisa dos resultados, porque além de acrescentar no resultado do plano o tempo inicial, o tempo final e custos estimados já apresentados no comando EXPLAIN na forma simples, apresenta também o tempo real de cada etapa do plano de execução.

Uma das melhorias em relação à versão simples é que com a utilização do ANALYZE é possível observar que os valores "actual time" são disponibilizados em milissegundos de tempo real, enquanto que os tempos estimados de custo (cost) são apresentados em unidades de pesquisa nas páginas de dados. Na variação mais sofisticada do comando, o tempo total de execução (total runtime) é o cálculo somado dos tempos de inicialização, finalização e processamento do resultado, deixando de lado os tempos de análise da consulta, bem como o tempo de reescrita.

Diferentemente do comando SELECT, a execução da instrução EXPLAIN ANALYZE para os comandos INSERT, UPDATE e DELETE pode apresentar um tempo de execução maior. Isso porque para executar qualquer um desses comandos é necessário antes de tudo que o SGBD realize uma consulta para verificar a localização dos registros a fim de serem inseridos, modificados ou excluídos. A **Figura 5** exemplifica o plano de execução para a exclusão de registros através do método EXPLAIN ANALYZE.

A funcionalidade do comando ANALYZE não se restringe ao maior detalhamento do plano de execução que é formulado pelo planejador do SGBD. Ele também atua para que haja a atualização das tabelas de

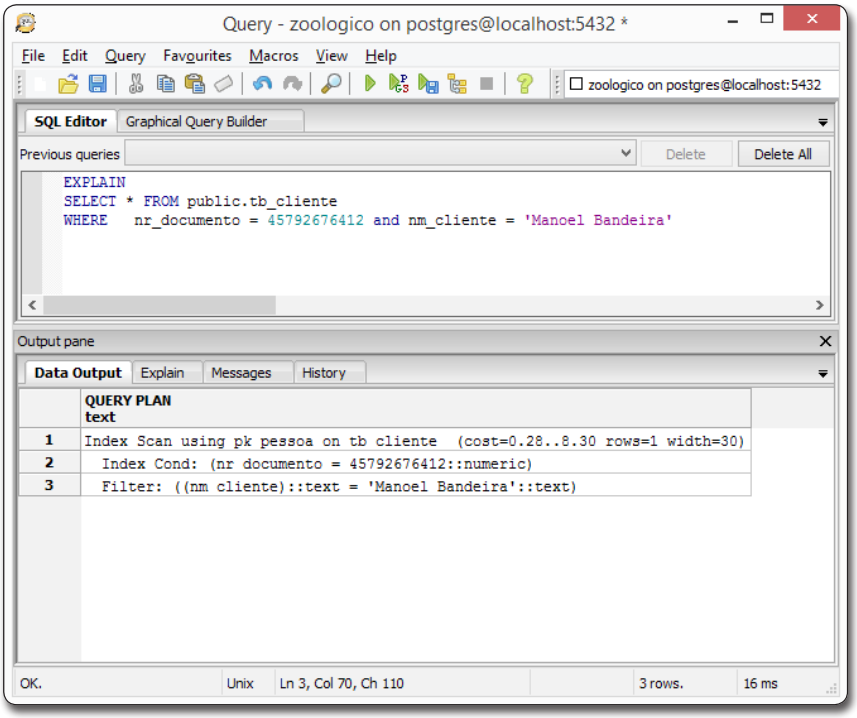



**Figura 4**. Exemplo de plano de execução com varredura utilizando índices

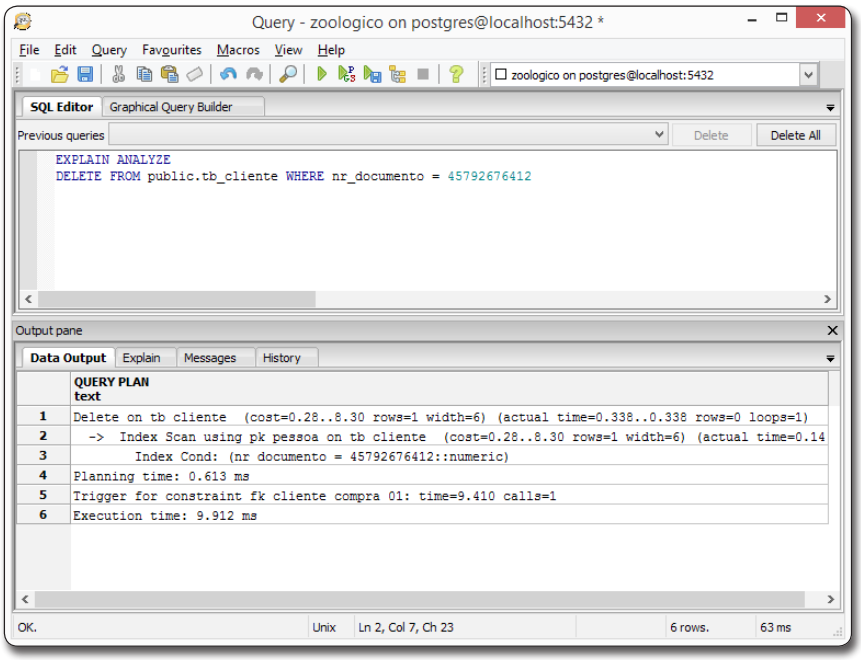



**Figura 5**. Exemplo do comando EXPLAIN ANALYZE para exclusão de registros

sistema responsáveis por armazenar informações que servirão de base para o planejador do PostgreSQL. Quando executado, o comando ANALYZE faz a readequação das estatísticas sobre as informações de distribuição dos dados na tabela pg_class. Caso essa atualização não aconteça, ou se a própria distribuição dos dados nas tabelas selecionadas na instrução SQL aumentarem ou diminuírem drasticamente, os cálculos em relação aos custos podem ser gerados de forma errônea, refletindo diretamente na eficiência dos planos de execução uma vez que esses serão calculados com base em dados desatualizados.

Além da variação ANALYZE do comando EXPLAIN, o planejador do SGBD também oferece a opção VERBOSE. Nessa variação o SGBD realiza uma apresentação detalhada da árvore do plano de execução, diferentemente das demais variações que mostram um resumo do passo a passo de consulta criado por ele. Apesar de retornar mais informações, ela não é muito usual entre os usuários do PostgreSQL por ter uma finalidade mais adequada para a depuração da query. A **Figura 6** demonstra o resultado dessa variação sobre uma consulta que utiliza junções entre tabelas.

Para um melhor entendimento dos métodos que o SGBD utiliza para realizar a varredura de registros nos blocos de disco, a **Tabela 1** mostra um resumo das principais operações do PostgreSQL utilizadas pelo planejador para a criação dos planos de consulta.

## Algumas boas práticas no PostgreSQL

Além dos conceitos de análise no processo de otimização de queries destacados durante o artigo, algumas boas práticas para a construção de instruções SQL podem ser refletidas e levadas em consideração para que os resultados obtidos tenham uma maior eficiência.

É bem comum que os usuários do banco utilizem um padrão de criação de consultas SQL onde é empregada toda a estrutura de colunas que desejam obter como resultado sobre um único comando SELECT, mesmo se houver uma alta complexidade da estrutura para processamento. Naturalmente, quanto mais forem as operações e métodos empregados em uma única instrução SELECT, maior será a sua complexidade, pois nesse padrão o SGBD se obrigará a realizar várias operações simultâneas, como filtros, ordenação, agregações, cálculos, entre outros.

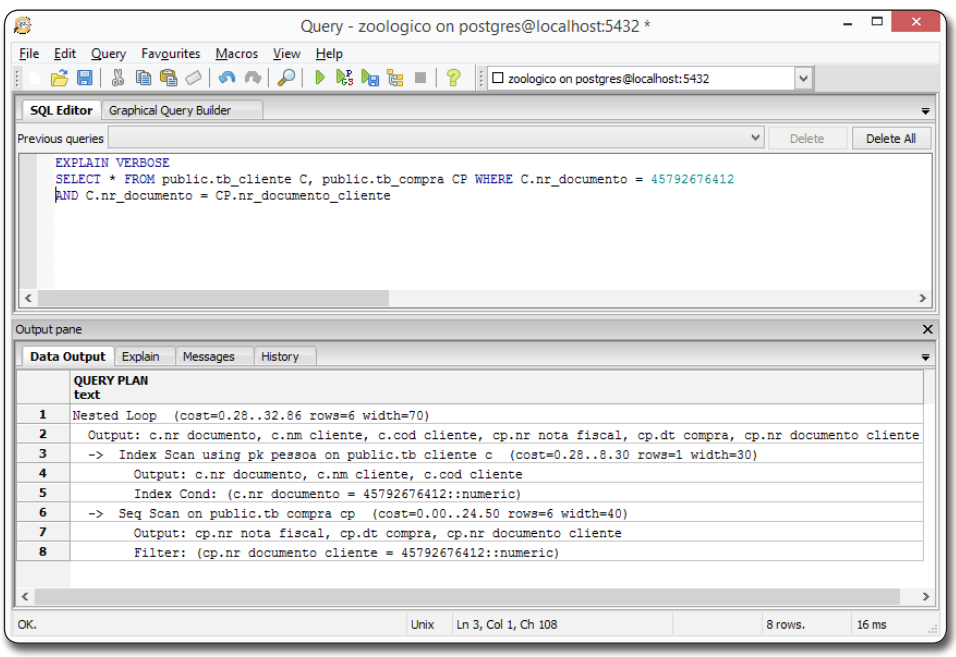



**Figura 6**. Exemplo do comando EXPLAIN VERBOSE

Um único comando SELECT pode ser considerado uma excelente técnica, porém pode atrapalhar quando o assunto for manutenção de código e emprego de otimizações. Dividir as sentenças em mais de um SELECT para se obter o relatório desejado pode ser a decisão mais sensata a longo prazo.

Outra prática saudável para o banco de dados é quando houver a necessidade de realizar os comandos de alteração de

| Método | Descrição |
|---|---|
| Seq Scan | É considerado o método de varredura mais básico e está associado à leitura sequencial da tabela. Quando há uma Seq Scan em um plano de execução, a tabela da instrução SQL será varrida por completo, da primeira à última linha. |
| Index Scan | Essa operação é apresentada quando existe uma leitura de alguma estrutura de índice, com o intuito de evitar que seja realizada a varredura completa de uma tabela, ou quando o plano de consulta privilegia a ordem apresentada nos índices para evitar uma operação Sort. |
| Sort | Acontece principalmente quando há um comando de ordenação explícito na instrução SQL, através do ORDER BY ou quando outra operação de plano de execução precisa de uma porção de dados ordenados para ser executada. |
| Unique | Essa operação é exclusiva para realizar a eliminação de registros repetidos em tabelas, principalmente quando há a presença de cláusulas de DISTINCT. |
| Nested Loop | A operação Nested Loop é essencial para a junção entre duas tabelas. Ela é acionada principalmente quando há a presença de INNER JOINs, LEFT JOINs e UNIONs. |
| Hash e Hash Join | Os métodos de Hash e Hash Join possuem comportamento similar aos presentes no Nested Loop, com a peculiaridade de não necessitar que as colunas participantes da junção estejam ordenadas. |

**Tabela 1**. Descrição dos principais métodos apresentados nos planos de consulta

registros (INSERT, UPDATE ou DELETE), os mesmos devem ser realizados em pequenos lotes. Esse método ajudará a obter um bom desempenho desses comandos, já que internamente o SGBD poderá preencher os arquivos de log de forma mais adequada, bem como ter uma melhor administração da quantidade de LOCKS (registros do mecanismo de controle de concorrência) exercidos na instrução SQL. Não há uma quantidade específica para limitar um bom desempenho, esse número é uma variável que dependerá do ambiente onde se está sendo trabalhado.

Esta técnica de otimização de resultados nos bancos PostgreSQL está relacionada à comparação de dados quando houver filtros na cláusula WHERE ou em relacionamentos de junções. Sempre é indicado que toda interação de comparação e de similaridade de registros, independentemente de serem do tipo inteiro, caractere etc., seja realizada em seu estado natural de tipo de dado, sem que haja conversões ou CASTs.

Em resumo, evite conversões de tipos de dados. Esses métodos, quando utilizados em alterações de tipos de dados em instruções SQL, oneram, e muito, o tempo de execução das queries.

Em resumo, podemos elencar dois itens essenciais para termos queries mais saudáveis para execuções no banco de dados. São eles:

1 - entender do ambiente no qual seus dados estão sendo manipulados;

2 - ter consciência do contexto da técnica que está sendo aplicada.

Existem diversas maneiras para melhorar e manipular instruções SQL para se buscar um resultado que satisfaça as pretensões do usuário final. Evidenciamos neste artigo alguns recursos importantes pertencentes ao PostgreSQL que podem ser aplicados em todos os tipos de aplicações, sejam elas de grande ou pequeno porte.

## Autor

**Thiago Pires**
*thiago.serip@gmail.com – http://medium.com/@thiagopires*
Entusiasta e em constante aprendizado no mundo dos bancos de dados. Bacharel em Ciência da Computação pela Universidade Luterana do Brasil. Possui alguns anos de experiência em desenvolvimento de sistemas e administração de banco de dados. Atualmente é DBA do Grupo API.

## Você gostou deste artigo?

Dê seu voto em ***www.devmedia.com.br/sqlmagazine/feedback***

Ajude-nos a manter a qualidade da revista!