

A Importância do Comando EXPLAIN ANALYSE em Sistemas de Gerenciamento de Bancos de Dados

Professor: Jorge Baldez

Introdução

Dentro do universo dos sistemas de gerenciamento de bancos de dados (SGBDs), como PostgreSQL, a ferramenta EXPLAIN ANALYSE desempenha um papel crucial na otimização e eficiência das consultas SQL.





Introdução

- O comando EXPLAIN e sua variação EXPLAIN ANALYZE são amplamente suportados em diversos Sistemas de Gerenciamento de Banco de Dados (SGBDs), cada um com suas próprias nuances e implementações. Aqui estão alguns dos SGBDs que utilizam essas ferramentas para análise e otimização de consultas
-

Introdução

- **PostgreSQL:**

- Suporta tanto EXPLAIN quanto EXPLAIN ANALYZE.
- EXPLAIN mostra o plano de execução estimado para uma consulta, enquanto EXPLAIN ANALYZE executa a consulta e exibe o plano real de execução, incluindo tempo de execução e outras estatísticas detalhadas.

- **MySQL:**

- Suporta o comando EXPLAIN, que exibe o plano de execução estimado da consulta.
- A partir das versões mais recentes, o MySQL também permite um nível detalhado de análise com EXPLAIN FORMAT=JSON, o que oferece uma visão mais profunda sobre como a consulta será executada.



Introdução

- **MariaDB:**

- MariaDB, um fork do MySQL, também suporta o EXPLAIN e o EXPLAIN FORMAT=JSON.
- Além disso, MariaDB oferece o ANALYZE FORMAT=JSON para fornecer insights detalhados sobre o desempenho da consulta.

- ☐ **Oracle Database:**

- Oracle oferece o comando EXPLAIN PLAN FOR para visualizar o plano de execução de uma consulta.
 - O comando é complementado pela exibição de planos de execução via ferramentas de monitoramento e análise do Oracle, como o SQL*Plus.
-



Introdução

- **SQL Server:**
 - No Microsoft SQL Server, EXPLAIN não é utilizado diretamente, mas o equivalente é o comando SET STATISTICS PROFILE ON ou SET SHOWPLAN_ALL ON, que fornece planos estimados ou reais de execução.
 - Há também a opção de visualizar os planos de execução nas ferramentas gráficas, como o SQL Server Management Studio (SSMS).
 - **SQLite:**
 - SQLite suporta o comando EXPLAIN, que fornece uma visão em nível de instrução sobre o que o banco de dados planeja fazer.
 - O comando EXPLAIN QUERY PLAN oferece uma visualização simplificada do plano de execução.
-

O que é EXPLAIN ANALYSE?

- **EXPLAIN ANALYSE** é um comando SQL usado para analisar o plano de execução de consultas SQL. Ele executa a consulta e fornece estatísticas detalhadas sobre o tempo e recursos utilizados, indo além do que o EXPLAIN simples oferece.
- **Quando Utilizar**
 1. Análise de Desempenho de Consultas
 2. Otimização de Consultas
 3. Indexação Eficaz
 4. Depuração



Exemplos de Comandos e Interpretação dos Resultados

Exemplo 1: Consulta Simples

- **Comando:**
- `EXPLAIN ANALYSE SELECT * FROM clientes WHERE id = 101;`
- **Resultado:**
- Seq Scan on clientes (cost=0.00..11.50 rows=1 width=240) (actual time=0.013..0.014 rows=1 loops=1) Filter: (id = 101) Rows Removed by Filter: 99 Planning Time: 0.052 ms Execution Time: 0.045 ms
- **Interpretação:**
- **Seq Scan:** Indica que foi feita uma varredura sequencial na tabela clientes.
- **Cost:** Estima o custo da operação (custo inicial..custo total).
- **Rows:** Número de linhas que o PostgreSQL espera processar.
- **Actual time:** Tempo real de execução (início..fim).
- **Rows Removed by Filter:** Quantidade de linhas descartadas pelo filtro.
- **Planning/Execution Time:** Tempo de planejamento e execução da consulta.

Exemplo 2: Consulta com Join

- **Comando:**
- `EXPLAIN ANALYSE SELECT * FROM pedidos JOIN clientes ON pedidos.cliente_id = clientes.id;`
- **Resultado:**
- Hash Join (cost=4.27..8.31 rows=100 width=384) (actual time=0.026..0.027 rows=100 loops=1)
Hash Cond: (pedidos.cliente_id = clientes.id)
-> Seq Scan on pedidos (cost=0.00..2.00 rows=100 width=148) (actual time=0.006..0.007 rows=100 loops=1)
-> Hash (cost=2.20..2.20 rows=120 width=236) (actual time=0.012..0.012 rows=120 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 26kB
-> Seq Scan on clientes (cost=0.00..2.20 rows=120 width=236) (actual time=0.003..0.005 rows=120 loops=1)
Planning Time: 0.065 ms
Execution Time: 0.076 ms

Interpretação

- **Hash Join (cost=4.27..8.31 rows=100 width=384) (actual time=0.026..0.027 rows=100 loops=1)**

O PostgreSQL está realizando um Hash Join entre duas tabelas, pedidos e clientes. O cost=4.27..8.31 indica o custo estimado dessa operação, começando de 4.27 e indo até 8.31. rows=100 sugere que o otimizador espera que 100 linhas sejam retornadas, e width=384 indica o tamanho médio de cada linha em bytes. Na prática, essa operação levou de 0.026 a 0.027 milissegundos (actual time) e foi executada uma vez (loops=1), retornando 100 linhas (rows=100). O PostgreSQL juntou as tabelas pedidos e clientes rapidamente, esperando encontrar cerca de 100 linhas, o que realmente aconteceu.

Interpretação

- **Hash Cond: (pedidos.cliente_id = clientes.id)**

Esta é a condição do join. O PostgreSQL usou o campo cliente_id da tabela pedidos e o campo id da tabela clientes para juntar as duas tabelas.

- **Seq Scan on pedidos (cost=0.00..2.00 rows=100 width=148) (actual time=0.006..0.007 rows=100 loops=1)**Aqui, o PostgreSQL fez uma varredura sequencial (Seq Scan) na tabela pedidos. Estimou um custo de 0.00 a 2.00, esperando encontrar 100 linhas (rows=100) com um tamanho médio de linha de 148 bytes (width=148). Na prática, essa varredura levou de 0.006 a 0.007 milissegundos e encontrou 100 linhas.
- **Hash (cost=2.20..2.20 rows=120 width=236) (actual time=0.012..0.012 rows=120 loops=1)**Este processo levou exatamente 0.012 milissegundos (actual time) e foi realizado uma vez (loops=1), processando 120 linhas (rows=120) como previsto.

Interpretação

- **Buckets: 1024 Batches: 1 Memory Usage: 26kB** Estes detalhes referem-se à operação de hash mencionada acima.
- **Buckets: 1024:** indica o número de compartimentos utilizados na estrutura de hash.
- **Batches: 1** significa que a operação de hash foi realizada em um único lote.
- **Memory Usage: 26kB:** mostra a quantidade de memória utilizada para a operação de hash.
- **Planning Time: 0.065 ms** Este é o tempo que o PostgreSQL levou para planejar a execução da consulta. Neste caso, foram 0.065 milissegundos.
- **Execution Time: 0.076 ms** Este é o tempo total que o PostgreSQL levou para executar a consulta, incluindo o join e as varreduras nas tabelas. O tempo foi de 0.076 milissegundos.

Melhores Práticas

- Use em ambiente de testes.
- Compare os planos de execução antes e depois das otimizações.
- Utilize como forma de definir melhor estratégia para suas consultas, definindo pela melhor performance.
- Realize análises periódicas.

Conclusão

- O EXPLAIN ANALYSE é essencial para a otimização de consultas em SGBDs. A compreensão dos resultados pode levar a melhorias significativas no desempenho, tornando-se uma ferramenta valiosa para profissionais da área de dados.