Processamento e Otimização de Consulta

Henrique Rodrigues Araújo

Sumário

1. Introdução

Quando a estrutura do banco de dados é pouco flexível , as operações sobre o banco de dados são variadas e existem diversas maneiras de se executar uma transação. O processamento e otimização de consulta é uma forma encontrada para amenizar os efeitos dessas ações sobre o banco de dados.

O objetivo desse trabalho e analisar qual são as ações no Sistema Gerenciador de Banco de Dados ORACLE usa como padrão para processamento e otimização de consulta e quais são as opções de alteração ele disponibiliza para o operador do banco.

2. Desenvolvimento

A otimização de consulta pode ser feita de três maneiras de forma baseado em regras heurísticas, baseado em estimativas de custo e a combinação das duas formas.

A forma de regras heurística usa de regras predefinidas para escolher o plano de acesso e muda a ordem de execução de operações baseada na equivalência de expressões da álgebra relaciona.

A forma de estimativas de custo usa a comparação de custo entre várias estratégias de execução e escolha a estratégia de menor custo.

1) RBO(Rule Based Otimizer)

Esta forma de otimização considera as regras de negócio para escolher a forma de recuperar as informações do banco de dados. A partir da versão 10g, a Oracle deixou de dar suporte ao otimizador, deixando suas funcionalidades disponíveis apenas para manter a compatibilidade com aplicações antigas. As novas aplicações desenvolvidas devem utilizar o CBO. Atualmente não são realizadas evoluções neste otimizador, mas apenas as correções de erros que eventualmente são encontrados.

Para melhorar o desempenho de uma consulta, o RBO verifica apenas uma maneira de otimização. Ao encontrar a primeira forma aplicável, ele abandona o processo sem verificar se outros mecanismos podem ser aplicados.

Esta série de artigos não detalhará este otimizador, uma vez não é recomendada a construção de novos sistemas que utilizem este otimizador para melhorar o desempenho.

Rank do RBO

caminho 1: Linha única por Rowid.

caminho 2: Linha única por junção de cluster.

caminho 3: Linha única por chave de cluster hash com chave única ou primária.

caminho 4: Linha única por chave exclusiva ou primária.

caminho 5: Junção em cluster

caminho 6: Chave de cluster com hash.

caminho 7: Chave de cluster indexada.

caminho 8: Indexe comosto.

caminho 9: Indexe de coluna única.

caminho 10: Pesquisa em intervalos limitados em colunas indexadas.

caminho 11: Pesquisa em intervalos ilimitados em colunas indexadas.

caminho 12: Sort Merge Join.

caminho 13: MAX ou MIM da coluna indexada.

caminho 14: ORDER BY na coluna indexada.

caminho 15: Verificação completa da tabela.

2) CBO (Cost-Based Optimizer)

Executa o comando de forma que consuma o mínimo de recursos de processamento. Para isto, o servidor de banco de dados busca maneiras alternativas para escrever o mesmo comando de forma que sua execução seja um processo mais simples. Para tentar verificar a melhor forma de escrever um comando, o otimizador utiliza as estatísticas e histogramas existentes para os objetos e operadores utilizados no comando. Caso as estatísticas ou histogramas não estejam disponíveis, o otimizador recorre a parâmetros previamente definidos para tentar chegar a uma solução melhor do que a apresentada.

Enquanto o RBO para a verificação logo após encontrar a primeira otimização possível, o CBO realiza todas as otimizações possíveis aplicáveis à consulta. O trabalho deste otimizador é maior do que o realizado pelo RBO, mas o resultado final é melhor do que o alcançado pelo otimizador baseado em regras de negócio, uma vez que os recursos gastos durante a execução da consulta são menores.

Este otimizador também permite que o desenvolvedor dê sugestões (hints) a respeito de qual é a melhor forma de resolver um comando. Esta dica pode ser, por exemplo, a indicação de qual a melhor maneira de acessar os dados em uma tabela, ou a forma mais vantajosa de efetuar a junção entre duas tabelas.

Disponível desde o Oracle 7, este é o principal otimizador do Oracle desde então, e a partir do Oracle 10g, tornou-se o único otimizador ainda em desenvolvimento. Por este motivo, este otimizador será descrito em detalhe nos próximos artigos.

3) Hints (sugestões)

O Oracle oferece hints(sugestões) que você pode especificar em uma determinada consulta para tentar conseguir melhorar o desempenho. Hints são utilizadas para influenciar o otimizador baseado em custo para controlar os métodos de acesso e condições de junções etc.

Segue abaixo a lista das hints mais usadas:

- **first_rows**: Para forçar o uso de índice de modo geral. Faz com que o otimizador escolha um caminho que apanha a 1º linha ou N linhas mais rapidamente.

- **all_rows**: Para forçar um scan complete na tabela.

- **full**: Para forçar um scan completo na tabela. O custo da leitura do índice e das linhas pode ser maior do que simplesmente ler a tabela inteira. A hint full também pode causar resultados inesperados como scan na tabela em ordem diferente da ordem de acesso.

- **index**: Para forçar o uso de um índice.

- **no_index**: É utilizado para evita que um índice especificado seja usado.

- **index_join** : Permite mesclar índice em uma única tabela. Permite acessar somente os índices da tabela, e não apenas um scan com menos bloco no total, é mais rápido do que usar um índice que faz scan na tabela por rowid.;

- **and_equal** : Para acessar todos os índices que você especificar. A hint and_equal faz com que o otimizador misture vários índices para uma única tabela em vez de escolher qual é ao melhor.

- **index_combine**: É utilizado para acessar diversos índices do tipo bitmap. Faz com que o otimizador misture vários índices bitmap para uma única tabela em vez de

- **index_ffs**: Força um scan completo do índice. Acessa apenas o índice, e não apenas a tabela correspondente. Ele só será usado se todas as informações que a consulta precisa apanhar estiverem no índice. Essa hint pode oferecer grandes ganhos de desempenho, especialmente quando a tabela também possuir um grande numero de colunas.

4) Variáveis Bind

Toda instrução SQL submetida ao banco de dados Oracle é colocada em cache. Uma instrução SQL colocada no cache é reutilizada se uma instrução idêntica é enviada para o banco de dados. Quando ocorre a reutilização de uma instrução, o tempo de execução é reduzido, pois o plano de execução já está traçado, não havendo necessidade de refazer o Parse. Entretanto, a instrução SQL deve ser absolutamente idêntica para ser reutilizada. Isso significa que:

- Todos os caracteres na instrução SQL devem ser iguais.
- Todas as letras na instrução SQL devem ter a mesma caixa.
- Todos os espaços na instrução SQL devem ser iguais.

Você pode garantir que uma instrução seja idêntica utilizando variáveis bind para representar valores de coluna. Elas funcionam como parâmetros em instruções SQL, possibilitando a atribuição de valores dinâmicos nos comandos SELEC, UPDATE, DELETE e INSERT. Com a utilização de variáveis Bind o Oracle faz o reuso de instrução SQL já armazenada em memória. Dessa forma, variáveis bind servem de ponte para a execução de uma instrução SQL, já preparada na memória do servidor, onde são enviadas apenas os valores nela contido.

3. Conclusão

O Sistema Gerenciador de Banco de Dados ORACLE, ja vem como dois otimizadores pensando de forma heurística e na forma e custo, mas a suas ultimas verções ele passou a atualizar so o otimizador de custo e manter o heurístico para evitar erros de compatibilidade.

O otimizador na forma de custo CBO ja é robusto com as diversas estatisticas que o ORACLE consegue gerar e ainda permite que o administrador do banco de susgestoes nas pesquisas dizendo qual o e camino a ser tomado para determinada pesquisa.

4. Referencia

Oracle Database, 2019.Disponível em: <https://docs.oracle.com/cd/B10500_01/server.920/a96533/rbo.htm#38960>. Acesso em:27 de novembro. de 2019.