**Candidato: Roberth Silva**

**roberth410@gmail.com**

**98 988529825**

**Prova Técnica da Coordly**

**Escreva uma query que otimize a seguinte consulta para grandes volumes de dados. Query original: *SELECT \* FROM Orders WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31' ORDER BY CustomerID;***

Solução:

* Evitar usar SELECT \* para reduzir a carga de I/O. Nesse caso estou supondo que o nome das colunas da tabela sejam: OrderID, CustomerID, OrderDate, TotalAmount
* Considerar a utilização do WITH (INDEX(...)) caso existe algum índice relevante na tabela.
* Considerar a substituição do comando BETWEEN para >= e < para evitar problemas com registros que possuem frações de segundo.

Dessa forma, a query otimizada ficaria no seguinte formato:

SELECT OrderID, CustomerID, OrderDate, TotalAmount

FROM Orders WITH (INDEX(Indice\_Orders))

WHERE OrderDate >= '2024-02-05' AND OrderDate < '2025-02-05'

ORDER BY CustomerID;

**Qual é a diferença entre INNER JOIN, LEFT JOIN e CROSS JOIN?**

Solução:

A diferença entre os três comandos estão na forma como eles associam/combinam os registros entre duas tabelas.

Inner Join – Junção Interna

Retorna apenas linhas que possuem correspondências em ambas as tabelas

SELECT Orders.OrderID, Customers.CustomerName

FROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

Nesse caso, só irá retornar items Orders que tem um cliente correspondente em Customers.

Left Join – Junção externa esquerda

Retorna todas as linhas da tabela da esquerda e apenas as correspondentes da tabela da direita. Se não houver correspondência, retorna um valor nulo nas colunas da tabela direita

SELECT Orders.OrderID, Customers.CustomerName

FROM Orders

LEFT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

A query acima vai retornar todos os pedidos, mesmo que não tenham um cliente associado. Se um pedido não tiver cliente correspondente, a coluna CustomerName será null.

Cross Join – Junção cruzada

Combina todas as linhas da primeira tabela com todas as linhas da segunda tabela, tal como um produto cartesiano.

SELECT Orders.OrderID, Customers.CustomerName

FROM Orders

CROSS JOIN Customers;

Precisa ser usado com cautela, pois gera uma grande quantidade de dados. Para a query acima, se a tabela Order poussuir 2000 registros e a tabela Customers possuir 5000 registros, o resultado será um retorno de 10 000 linhas.

**Explique a diferença entre os modos de carregamento Lazy, Eager e Explicit no Entity Framework.**

No Entity Framework os modos de carregamento definem como as entidades relacionadas são carregadas no contexto do banco de dados.

No modo Lazy (Lazy loading), os dados das entidades relacionadas são carregados automaticamente quando acessados pela primeira vez, mas não são incluídos na consulta inicial. Tem como vantagem reduzir o tráfego de dados inicialmente, carregando entidades somente quando necessário, no entanto, uma desvantagem é de que pode gerar várias consultas SQL adicionais.

No modo Eager (Eager Loading), as entidades relacionadas são carregadas imediatamente na consulta inicial, usando a instrução Include(). Tem como vantagem reduzir o número de consultas ao banco de dados, carregando os dados de uma vez só. Tem como desvantagem a possibilidade de trazer mais dados do que o necessário, aumentando o consumo de memória e processamento.

No modo Explicit (Explicit Loading), os dados das entidades relacionadas não são carregados automaticamente, mas podem ser carregados manualmente através das instruções context.Entry().Collection().Load() ou context.Entry().Reference().Load(). A vantagem é de que permite controle total sobre quando e quais dados carregar,e a desvantagem é de que exige um código a mais para carregar manualmente os dados.

**Descreva um cenário onde o uso de AsNoTracking() seria necessário e explique por que.**

Um cenário ideal para usar AsNoTracking() é quando se precisa apenas ler dados sem a necessidade de modificá-los, como em consultas de relatórios ou dashboards.

Ex: quando se tem uma API que exibe dados de listagem de produtos, sem precisar alterá-los.

Tem como vantagem a melhora do desempenho, redução o consumo de memória e evita conflitos pois não tenta detectar mudanças nos items carregados.

**Explique a diferença entre arquitetura monolítica e arquitetura de microsserviços.**

Na arquitetura monolítica, o sistema é desenvolvido como uma única aplicação onde todos os componentes (UI, lógica de negócios, acesso a dados) estão integrados.

* Código centralizado em um único projeto.
* Todos os módulos compartilham o mesmo banco de dados.
* Facilidade de desenvolvimento e implantação inicial.
* Escalabilidade vertical (aumentar recursos do servidor).

Podemos citar como desvantagens:

* Difícil de escalar independentemente.
* Manutenção complexa conforme cresce.
* Qualquer falha pode afetar toda a aplicação

Na arquitetura de microsserviços, o sistema é dividido em múltiplos serviços independentes, cada um responsável por uma funcionalidade específica e se comunicando principalmente via APIs.

Tem como características:

* Cada serviço tem sua própria lógica e pode ter banco de dados separado.
* Permite escalabilidade horizontal (escalar serviços específicos conforme a demanda).
* Facilita a manutenção e atualização sem afetar todo o sistema.
* Maior flexibilidade para usar tecnologias diferentes em cada serviço.

Exemplo:

No e-commerce, serviços separados para login, catálogo de produtos e pagamentos, podendo ser escritos em diferentes linguagens e escalados separadamente.

Também pode ter as seguintes desvantagens:

* Maior complexidade na comunicação entre serviços.
* Requer uma boa estratégia de orquestração e monitoramento.
* Mais difícil de depurar e gerenciar falhas.