**Estudo aprofundado sobre Fetch Types: Lazy e Eager**

Aprendemos que geralmente encontramos dois tipos de carregamento de dados: "lazy" e "eager". Esses dois conceitos são essencialmente sobre quando e como os dados são recuperados do banco de dados para serem usados em nossas aplicações.

**O que é Fetch Type?**

De modo bem simples, Fetch Type define qual a estratégia será utilizada para carregar os dados do banco para sua aplicação.

Para facilitar nosso entendimento, gosto de usar a analogia do café da manhã. Imagine que você tem uma mesa de café da manhã e pode haver vários itens nela, como pão, café, leite, frutas, etc.

A forma como você vai pegar esses itens, quando e quantos de uma vez, é basicamente o conceito por trás do fetch type, que veremos a seguir.

**Lazy Fetch Type**

Lazy, em inglês, significa preguiçoso. Em termos de programação, Lazy Fetch Type é quando você pega apenas o que precisa, no momento em que precisa.

Se voltarmos para a nossa analogia do café da manhã, seria como pegar somente o café primeiro. Quando se sentir pronto para comer algo, você então vai e pega uma fruta ou um pão. Ou seja, você só busca os outros itens quando realmente vai utilizá-los.

Vamos tomar como exemplo a relação entre um usuário e seus posts em uma aplicação de blog. Se optarmos pelo fetch type lazy, ao carregarmos o usuário, seus posts não serão carregados automaticamente. Eles serão postergados até que explicitamente solicitado, conforme a seguir:

@Entity

**public** **class** **User** {

@OneToMany(fetch = FetchType.LAZY)

**private** List<Post> posts;

}

COPIAR CÓDIGO

**Eager Fetch Type**

Eager, em inglês, pode ser traduzido como ansioso. Em programação, Eager Fetch Type é mais rápido, pois vai pegar todos os dados relacionados ao mesmo tempo.

Back to the café da manhã analogia, Eager Fetch Type seria como se você pegasse tudo que há na mesa de uma vez só. Você pega o café, pão, leite, frutas, tudo de uma única vez.

Na relação usuário/posts, se optarmos pelo fetch type eager, ao carregar o usuário, todos os seus posts serão carregados simultaneamente.

@Entity

**public** **class** **User** {

@OneToMany(fetch = FetchType.EAGER)

**private** List<Post> posts;

}

COPIAR CÓDIGO

**Impacto no desempenho da aplicação**

A estratégia de carregamento afeta diretamente o desempenho da aplicação. Um carregamento Eager pode parecer eficiente, pois tudo já está carregado de uma vez. No entanto, se a relação envolver muitos dados, isso pode causar problemas de desempenho, além de consumir muito mais memória, uma vez que estamos carregando mais dados do que realmente precisamos.

Por outro lado, Lazy Fetch Type pode parecer uma estratégia mais eficaz, pois carrega os dados sob demanda. No entanto, se não administrado cuidadosamente, pode acabar resultando em múltiplas chamadas ao banco de dados, aumentando o tempo de resposta.

Escolher entre Lazy e Eager não é uma decisão trivial e deve ser baseada na necessidade da aplicação. Um bom ponto de partida é começar com Lazy Fetch Type e optar por Eager onde o carregamento completo é muitas vezes necessário.

O escopo da aplicação, a quantidade de dados, a frequência de acesso e muitos outros fatores serão decisivos para essa escolha. É importante sempre analisar o contexto e testar o desempenho para alcançar a melhor estratégia.

Entender sobre os conceitos de Fetch Types, lazy e eager, é um passo muito importante para melhorar o desempenho do seu aplicativo. Portanto, sempre busque melhorar e aperfeiçoar seu conhecimento nesta área.