Quando começamos a aprender sobre o desenvolvimento web com Java, em algum momento, nos deparamos com a necessidade de interagir com bancos de dados. Dentro deste universo, podemos fazer vários tipos de busca, como as derived queries, a JPQL (Java Persistence Query Language) e o próprio SQL nativo.

Mas qual é a diferença entre elas e como isso afeta a portabilidade entre diferentes bancos de dados? Iremos analisar aqui as queries nativas e a JPQL.

A JPQL é uma linguagem de consulta orientada a objetos que foi definida como parte da especificação JPA (Java Persistence API). Ela é usada para fazer consultas em bancos de dados relacionais de maneira similar ao SQL, mas com uma diferença fundamental: em vez de trabalhar com tabelas e colunas, como no SQL, a JPQL trabalha com classes e atributos que fazem parte do seu modelo de domínio.

// Exemplo de Query JPQL

**String** jpql = "select c from Customer c where c.name like :name";

COPIAR CÓDIGO

**2. Entendendo o SQL Nativo**

O SQL Nativo, por outro lado, é a linguagem padrão usada para interagir com bancos de dados relacionais. Com ela, você escreve consultas que são específicas para a estrutura e sintaxe do banco de dados que está usando.

-- Exemplo de Query SQL Nativo

**SELECT** \* **FROM** Customer **WHERE** name **LIKE** '%John%';

COPIAR CÓDIGO

Para utilizá-las no repositório, devemos usar o atributo nativeQuery da anotação @Query, conforme visto em vídeo.

A principal diferença entre JPQL e SQL Nativo é, portanto, o nível de abstração. A JPQL abstrai os detalhes do banco de dados, permitindo que você trabalhe no nível do modelo de domínio. Isso pode tornar o código mais legível e fácil de manter.

O SQL Nativo, por outro lado, permite um controle mais detalhado e direto sobre as consultas ao banco de dados. Isso pode ser útil se você precisa otimizar suas consultas para um banco de dados específico.

Porém, há a questão da portabilidade. Como a JPQL é uma abstração de alto nível, ela é compatível com qualquer banco de dados que suporte a especificação JPA. Se você precisar mudar seu aplicativo de um banco de dados para outro, a maioria das suas consultas JPQL continuarão funcionando sem alterações.

Já com o SQL Nativo, se você decidir mudar de um banco de dados para outro, provavelmente terá que reescrever uma parte das suas consultas, visto que cada banco tem suas particularidades de sintaxe e funcionamento.

A escolha entre JPQL e SQL Nativo muitas vezes depende das necessidades do seu projeto. Se a portabilidade é uma prioridade e se o seu modelo de domínio é complexo e rico de informações, a JPQL pode ser a melhor escolha. Se o desempenho é uma prioridade e você precisa de controle detalhado sobre suas consultas, o SQL Nativo pode ser a melhor opção.

Entender esses conceitos possibilitará que você tome decisões mais acertadas, minimizando problemas futuros e otimizando a performance de suas aplicações. Então, continue estudando e aprofundando seus conhecimentos!