### JDBC além do básico

Praticamente todos os sistemas que são desenvolvidos hoje em dia utilizam algum tipo de repositório lógico para armazenar as informações preenchidas pelos usuários.

Este passo é necessário, pois torna possível manter o estado do sistema entre suas falhas, reinícios e atualizações de hardware e software. Dentre estes tipos de repositórios, Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR) ainda se destacam como alternativa principal, dada a maturidade da tecnologia e seu formalismo matemático.

No Java, diversas especificações foram criadas para facilitar a transformação de objetos em um formato relacional, composto primariamente de tabelas, linhas e colunas.

Os documentos que definiram como utilizar EJB 1.x, EJB 2.x, JDO (agora sob responsabilidade da Apache) e, mais recentemente, JPA, têm em comum tirar do programador a necessidade de aprender dois mundos distintos (Java e SQL), introduzindo camadas intermediárias que fazem o chamado mapeamento objeto-relacional (MOR) de forma transparente.

Outra similaridade entre estas tecnologias reside no fato de que elas usam a API da *Java Database Connectivity* (JDBC) para realizar conexões com os SGBDs, enviar consultas, delimitar transações, etc.

De um modo geral, as ferramentas de mapeamento objeto-relacional realizam bons trabalhos, simplificando a vida dos desenvolvedores que desejam acessar SGBDs (apesar de fomentar a criação dos indesejados métodos de acesso nos *beans*, os conhecidos getXXXX() e setXXXX(...) – entretanto, este é um assunto para outro artigo).

Contudo, existem situações onde as ferramentas de MOR não conseguem obter um bom desempenho. A seguir, são descritos apenas alguns dos problemas mais comuns em projetos onde o JPA é utilizado:

* **Geração de consultas SQL que não utilizam índices:** qualquer SQL minimamente complexo a ser gerado (ex.: sub-consultas, junções externas, etc.) tem alta probabilidade de ser ineficiente;
* **Carregamento *batch*:** apesar de possuir mecanismos que atenuem este problema, a inserção de grande quantidade de dados no SGBD não é aconselhável com o JPA, uma vez que ela não foi projetada para essa finalidade;
* **Mapeamento de tabelas que possuem chaves compostas**: as chaves compostas sempre representaram um problema para o JPA. Quando puder evitá-las, faça-o;
* **Mapeamento do conceito de herança para o esquema relacional**: apesar de a herança ser uma característica muito poderosa da orientação a objetos, ela tem como ponto negativo o fato de aumentar consideravelmente o acoplamento entre as classes, o que dificulta muito a manutenção de um sistema.

Ainda assim, é muito comum ver desenvolvedores criando classes que herdam de outras de forma não planejada, empobrecendo muito o projeto final. O JPA prevê alguns meios de contornar este problema, com as técnicas de mapeamento (tabela única por hierarquia, uma tabela para cada classe da hierarquia, etc.).

Contudo, nenhum deles consegue satisfazer totalmente os requisitos de orientação a objetos e de normalização de dados. A melhor dica é estudar OO o suficiente para fugir da herança;

* **Geração de relatórios**: o JPA não foi projetado para utilização em ferramentas OLAP (*On-line Analytical Processing*), caindo novamente no problema de geração de consultas ineficientes;
* **Extenso número de artefatos de configuração a serem mantidos**: apesar de existirem ferramentas que mantêm o sincronismo entre a especificação das tabelas no SGBD e as anotações ou arquivos XML, é fácil perder o controle sobre esses artefatos conforme o projeto cresce.

Repare que alguns destes problemas são ocasionados pela diferença dos mundos relacional e orientado a objetos, outros são causados pela utilização da tecnologia para fins diferentes daqueles para os quais ela foi projetada.

De todo modo, é comum encontrar situações onde o JPA não se encaixa no seu projeto. Nestas situações, ao invés de perder um tempo precioso para aprender uma nova ferramenta, a utilização do bom e velho JDBC deve ser considerada.

Para isso, os desenvolvedores Java devem ter conhecimentos mais aprofundados sobre os passos que um banco de dados realiza ao processar uma consulta SQL, bem como de algumas funcionalidades existentes nos SGBDRs que facilitam a construção de sistemas.

O grande sucesso da JPA, entretanto, passou a esconder um problema na formação dos desenvolvedores: a maioria deles desconhece algumas funções extremamente importantes do JDBC, pois aprenderam somente o básico: abrir conexão com o SGBD, executar uma consulta, iterar sobre o resultado e fechar a conexão.