06

Evitando o N+1

Desejamos evitar o problema de ***N+1 queries***, otimizando nosso sistema. Para isso, criaremos um novo método, chamado `listaComProdutos` em nosso `CategoriasDAO`. Ele também retorna uma lista de categorias, mas o SQL que executa é um pouco diferente, efetuando o `join` com a tabela `Produto` e o *alias* dos campos:

```
public List<Categoria> listaComProdutos() throws SQLException {
    List<Categoria> categorias = new ArrayList<>();

    String sql = "select c.id as c_id, c.nome as c_nome, p.id as p_id, p.nome as p_nome, p.descricac
    try(PreparedStatement stmt = con.prepareStatement(sql)) {
        stmt.execute();
        try(ResultSet rs = stmt.getResultSet()) {
            while(rs.next()) {
                // implementar o laco
            }
        }
    }

    return categorias;
}
```

Precisamos implementar o laço que cria as categorias. Lembrando que o `join` traz a mesma categoria diversas vezes, primeiro declaramos a última categoria criada no começo do método:

```
List<Categoria> categorias = new ArrayList<>();
Categoria ultima = null;
```

Agora, dentro do laço `while`, do nosso `ResultSet`, leia os campos `c_id` e `c_nome` em variáveis `id` e `nome`. Feito isso, verificamos se a última categoria criada é nula (estamos na primeira iteração do laço) ou se o nome dela mudou, e nesse caso criamos uma nova categoria e adicionamos na lista:

```
if(ultima==null || !ultima.getNome().equals(nome)) {
    Categoria categoria = new Categoria(id, nome);
    categorias.add(categoria);
    ultima = categoria;
}
```

Fora do `if`, devemos ler os dados do produto. Leia os campos `p_id`, `p_nome` e `p_descricao`. Crie um novo produto, *sete* seu `id` e adicione-o, chamando o método `adiciona`, da classe `Categoria`.

Note que este método ainda não existe. Vamos na classe `Categoria` e adicionamos uma lista de produtos:

```
private final List<Produto> produtos = new ArrayList<>();
```

Adicione o *getter* desses produtos e o método `adiciona`, que adiciona um produto à lista.

Voltamos à classe de teste e mudamos a invocação, agora para o método `listaComProdutos`. Mude o laço interno dos produtos para, ao invés de invocar um `new ProdutosDAO`, invocar simplesmente o `categoria.getProdutos`.

Opinião do instrutor

A classe `CategoriasDAO` ficará assim:

```
public class CategoriasDAO {

    private final Connection con;

    public CategoriasDAO(Connection con) {
        this.con = con;
    }

    public List<Categoria> lista() throws SQLException {
        System.out.println("Executando uma query");
        List<Categoria> categorias = new ArrayList<>();

        String sql = "select * from Categoria";
        try(PreparedStatement stmt = con.prepareStatement(sql)) {
            stmt.execute();
            try(ResultSet rs = stmt.getResultSet()) {
                while(rs.next()) {
                    int id = rs.getInt("id");
                    String nome = rs.getString("nome");
                    Categoria categoria = new Categoria(id, nome);
                    categorias.add(categoria);
                }
            }
        }
        return categorias;
    }

    public List<Categoria> listaComProdutos() throws SQLException {
        List<Categoria> categorias = new ArrayList<>();
        Categoria ultima = null;

        String sql = "select c.id as c_id, c.nome as c_nome, p.id as p_id, p.nome as p_nome, p.descr
        try(PreparedStatement stmt = con.prepareStatement(sql)) {
            stmt.execute();
            try(ResultSet rs = stmt.getResultSet()) {
                while(rs.next()) {
                    int id = rs.getInt("c_id");
                    String nome = rs.getString("c_nome");
                    if(ultima==null || !ultima.getNome().equals(nome)) {
                        Categoria categoria = new Categoria(id, nome);
                        categorias.add(categoria);
                        ultima = categoria;
                    }
                    int idDoProduto = rs.getInt("p_id");
                    String nomeDoProduto =rs.getString("p_nome");
```

```
                    String descricaoDoProduto = rs.getString("p_descricao");
                    Produto p = new Produto(nomeDoProduto, descricaoDoProduto);
                    p.setId(idDoProduto);
                    ultima.adiciona(p);
                }
            }
        }
        return categorias;    }

}
```

E a classe de testes:

```
public class TestaCategorias {

    public static void main(String[] args) throws SQLException {
        try(Connection con = new ConnectionPool().getConnection()) {
            List<Categoria> categorias = new CategoriasDAO(con).listaComProdutos();
            for(Categoria categoria : categorias) {
                System.out.println(categoria.getNome());

                for(Produto produto : categoria.getProdutos()) {
                    System.out.println(categoria.getNome() + " - " + produto.getNome());
                }

            }
        }
    }

}
```