SnailDB: Banco de Dados Orientado a Objetos utilizando Prevayler

Giovane Roslindo Kuhn (FURB)
brain@netuno.com.br

Vitor Fernando Pamplona (FURB)
vitor@babaxp.org

Categoria: Banco de Dados.

Linguagem de programação: Java.

Sistema operacional: Qualquer com suporte a JRE 5.0.

Palavras-chave: DDL. DML. JDBC. Persistência. Prevayler. SGBDR. SGBDOO. SQL.

1 Contexto

A programação orientada a objetos já existe há muitos anos, mas somente a partir de 1998 começou a receber fortes investimentos internacionais e evoluções consideráveis nas pesquisas da engenharia. Nos últimos anos notou-se que a maioria das aplicações usa orientação a objetos (OO) para criar telas, regras de negócio e a sua arquitetura, mas continua mantendo os seus dados de maneira relacional, nos bancos de dados relacionais. Para a comunicação entre uma arquitetura OO e base de dados relacional, foram criados vários conceitos, técnicas e *frameworks*, todos eles baseados no conceito de mapeamento objeto/relacional.

A proposta do projeto SnailDB é criar um *framework* que permita que uma aplicação legada, ou já existente, tenha a sua fonte de dados alterada do modelo relacional para o orientado a objeto sem alterar uma única linha de código, como pode ser visto na fig. 1. Para isto, o *framework* permite definir, consultar e manipular coleções de objetos como se fossem tabelas de um banco, inclusive possuindo uma interface de comandos *Structured Query Language* (SQL). Assim, ao executar todos os *scripts* de criação da base de dados no SnailDB, é possível, aos poucos, retirar a parte relacional da aplicação legada, permitindo-a acessar todo o grafo de objetos criado pelo *framework* diretamente.

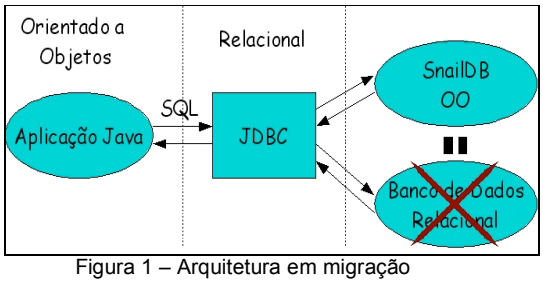

Figura 1 – Arquitetura em migração

Como conseqüência, a arquitetura do SnailDB também pode ser utilizada para agilizar a criação das consultas em coleções de objetos Java e permite uma manipulação em massa dessas coleções. A linguagem SQL poderá executada sobre *Lists*, *Maps* e outras estruturas existentes na linguagem.

O *framework* inicial tem as mesmas características de um SGBDR[1], isto é, trabalha com dados de forma relacional, via uma interface Java Database Connectivity (JDBC), e grava os dados de maneira orientada a objeto. Ele é capaz de interagir com qualquer aplicação feita em Java, como se fosse um banco de dados relacional comum.

---

1 Sistema Gerenciados de Banco de Dados Relacional

As características desenvolvidas para o SnailDB, em sua primeira versão, são as seguintes:

a) uma camada de persistência dos objetos utilizando a Application Programming Interface (API) de prevalência Prevayler (PREVAYLER TEAM, 2004). O Prevayler é um projeto brasileiro para persistência de objetos Java com extrema facilidade de uso. O grafo dos objetos em memória é persistido de maneira que, ao ser carregado, todas as referências dos objetos, suas dependências e associações continuem intactas;

b) uma linguagem para criação das classes e manipulação de dados em tempo de execução, seguindo as especificações da Data Definition Language (DDL) e Data Manipulation Language (DML) respectivamente. Os comandos implementados são o *CREATE TABLE*, que criará uma classe Java, e o *INSERT INTO*, que irá criar e preencher os objetos definidos pelo *CREATE TABLE*;

c) uma linguagem para consulta de objetos seguindo a especificação da SQL92. O comando *SELECT* deve ser implementado com possibilidade de junções e filtros.

2 Desenvolvimento

Inicialmente foi definida a BNF para a linguagem de criação, manipulação e seleção dos objetos no SGDB. O compilador para a linguagem é gerado com o JavaCC (JAVACC, 2005), um gerador de compiladores que utiliza a BNF definida como entrada.

O comando CREATE TABLE gera, fisicamente, o fonte de uma nova classe no banco. Este fonte é gerado com o auxílio do Velocity (APACHE FOUNDATION, 2005), um gerador de código baseado em *templates* e compilado com o compilador do Eclipse (ECLIPSE FOUNDATION, 2005), ambiente de desenvolvimento utilizado no projeto. Ao gerar a classe, o SnailDB popula uma estrutura de metadados, que é mantida junto com os futuros dados desta classe no Prevayler. Assim, quando a aplicação for reiniciada, a base de dados pode continuar com o mesmo estado da última execução.

As classes são geradas dinamicamente no SGBD, porém o *classloader*[1] padrão do Java não enxerga as classes geradas. Para resolver esta questão, foi implementado um *classloader* que conhece as configurações do SGBD e conseqüentemente a localização física das classes geradas.

O comando *INSERT INTO* cria uma nova instância da uma classe gerada pelo comando *CREATE TABLE*. Os objeto instanciados são persistidos no Prevayler para que posteriormente possam ser manipulados e selecionados pelo usuário.

O comando *SELECT* navega pelo grafo de objetos persistidos no Prevayler, selecionando os objetos que satisfazem as condições definidas no comando. No retorno estes objetos são convertidos para dados relacionais, permitindo o desenvolvedor manipulá-los através do JDBC.

Todos os comandos são implementados com o padrão de projeto *Visitor*, que tem o objetivo de navegar na árvore sintática e executar a respectiva operação. Um dos benefícios deste padrão é o desacoplamento da sintaxe da linguagem da forma com que os comandos são executados, com isso, mudanças na forma de execução dos comandos não afetam a árvore sintática da linguagem.

3 Resultados

Para a validação do *framework* foram desenvolvidos testes unitários simulando a execução de uma aplicação. Nos testes, a definição de classes simples, isto é, sem associações, puderam ser validadas, assim como a inserção e seleção de objetos. Outra implementação para validação foi um terminal de interação com o usuário, onde é possível escrever comandos para o SnailDB executar e então visualizar as respostas retornadas.

---

1 Mecanismo que o Java utiliza para carregar classes para a memória do programa

4 Referências

APACHE FOUNDATION. **Velocity**. [S.l.], 2005. Disponível em: <http://jakarta.apache.org/velocity/>. Acesso em: 14 out. 2005.

ECLIPSE FOUNDATION. **Eclipse**: main page. [S.l.], 2005. Disponível em: <http://www.eclipse.org/>. Acesso em: 14 out. 2005.

JAVACC. **Java compiler compiler**. [S.l.], 2005. Disponível em: <https://javacc.dev.java.net/>. Acesso em: 14 out. 2005.

PREVAYLER TEAM. **Prevalence skeptical FAQ**. [Curitiba], [2004]. Disponível em: <http://www.prevayler.org/wiki.jsp?topic=PrevalenceSkepticalFAQ>. Acesso em: 14 out. 2005.