VOCÊ AINDA USA BANCO DE DADOS? - UM QUESTIONAMENTO CÉTICO

Klaus Wuestefeld

Diretor de Desenvolvimento - Objective Solutions

EMail: klaus@objective.com.br Tel: (41) 254-3063

Resumo:
A prevalência de objetos é ordens de grandeza mais **rápida** e mais **simples** que usar um banco de dados tradicional para persistir objetos de negócios. Além disso, é robusta e transparente: persiste objetos **nativos do Java** e não requer alteração nas classes nem no ambiente de execução. Ou seja, anuncia o ocaso dos sistemas gerenciadores de bancos de dados.

Palavras-chave: prevalência, Java, objetos, persistência, transparência, DBMS

VOCÊ AINDA USA BANCO DE DADOS?

UM QUESTIONAMENTO CÉTICO

A prevalência de objetos é ordens de grandeza mais **rápida** e mais **simples** que usar um banco de dados tradicional para persistir objetos de negócios. Além disso, é robusta e transparente: persiste objetos **nativos da linguagem** e não requer alteração nas classes nem no ambiente de execução.

Como isso é possível?

Memória RAM está ficando cada vez mais barata. Para sistemas com centenas de megabytes ou até alguns gigabytes de dados, já hoje, é perfeitamente possível manter todos os objetos de negócios na RAM.

Seria fantástico poder manter todos meus objetos na RAM e esquecer a dor-de-cabeça de banco de dados. É isso que você está sugerindo?

Exatamente.

Você ficou maluco? E se meu sistema cair?

Para evitar a perda dos dados, basta salvar em arquivo, periodicamente, uma fotografia, um "snapshot" do estado do sistema. Essa operação é similar à hibernação de um notebook. A diferença é que, depois de tirado o "snapshot", seu sistema continua no ar.

Mas como ficam as transações executadas desde o último "snapshot"? Não se perdem com uma queda do sistema?

Não. Embora você precise de RAM suficiente para conter todos seus objetos de negócios, a persistência e integridade deles estão sempre garantidas.

Como?

Toda transação, antes de ser executada, é gravada em um arquivo de log. Depois de uma queda, o sistema é recuperado do último "snapshot" e as transações do log são simplesmente re-executadas. O sistema volta ao ar, então, exatamente no estado em que estava no momento da queda.

Quer dizer que meus objetos de negócios têm que ser determinísticos?

Isso mesmo. Depois de criados e dados a mesma seqüência de comandos, eles têm que produzir os mesmos resultados.

O sistema não tem que entrar em modo de somente leitura, durante a geração do "snapshot", para garantir sua consistência?

Não. Se o sistema não puder entrar em modo de somente leitura, nem durante alguns minutos na madrugada, o "snapshot" é gerado por uma réplica. Essa réplica lê do log todas as transações executadas no sistema "quente" e executa-as exatamente na mesma ordem. Em determinado momento, a réplica pára e seu "snapshot" é gerado com segurança. Depois disso, a réplica continua executando as transações do log, do ponto onde tinha parado, e volta a ficar em sincronismo com o sistema "quente", que sequer tomou conhecimento do processo.

Se o sistema cair, essa réplica não pode assumir?

Pode. Mencionei uma, mas você pode ter quantas réplicas quiser. Se o sistema "quente" cai, qualquer réplica pode ser eleita e assumir o sistema em poucos milisegundos. Naturalmente, você pode colocar cada réplica em uma máquina física diferente. No caso de uma queda, seu sistema não vai somente persistir, ele vai prevalecer.

Como ficam as regras de negócios envolvendo data e hora? As réplicas não perdem a sincronia?

Não. Do ponto de vista dos objetos de negócios, cada transação é executada exatamente no mesmo milisegundo no sistema "quente" e em todas as réplicas.

Esse esquema todo tem nome?

Tem. Chama-se prevalência de objetos. Abrange persistência, tolerância a falhas e balanceamento de carga transparentes para objetos de negócios.

Existe uma implementação Java que eu possa usar?

Existe: o Prevayler.

Quanto custa?

Nada. O Prevayler é software livre, licenciado sob a LGPL (Lesser General Public License) da Free Software Foundation.

Esse tal de Prevayler é rápido?

Em testes de carga, pesquisando um milhão de objetos por igualdade de atributo e retornando cem, sistemas usando o Prevayler são até dez mil vezes mais rápidos, isso mesmo: **DEZ MIL VEZES MAIS RÁPIDOS**, que sistemas usando, via JDBC, bancos de dados consagrados como o ORACLE e o MySQL. Quanto a processamento de transações, o Prevayler está equiparado aos bancos de dados tradicionais.

Você acha justo comparar acesso a memória com acesso a disco?

Os resultados mencionados acima foram atingidos mesmo com todas as tabelas dos bancos de dados "cacheadas" em memória! Ou seja, a diferença entre memória e disco nem está sendo levada em consideração.

Por que, então, a prevalência de objetos é tão mais rápida que usar um banco de dados tradicional?

Os objetos estão sempre em sua forma nativa. Não é necessária a comunicação entre processos nem conversão de dados em objetos. Não é necessário gerar nem processar comandos SQL. Não há necessidade de ganchos colocados no seu código por pré ou pós-processadores, como no caso dos bancos de dados OO. Não há restrição: você pode usar qualquer algoritmo ou estrutura de dados que sua linguagem suporte. É difícil ser mais rápido que isso.

Como é que o Prevayler gera os "Snapshots"?
Ele simplesmente serializa os objetos de negócios do sistema usando a serialização padrão do Java.

Quais são os padrões ou restrições que minhas classes de negócios têm que obedecer?
Elas têm que ser determinísticas, como disse acima, e serializáveis.

Só isso? Classes de negócios Java já não são determinísticas e serializáveis por natureza?
São.

Não existe mais nenhuma restrição para minhas classes de negócios?
Não. Isso é justamente o conceito da transparência.

Quais são as restrições para minhas classes cliente?
As classes cliente executam suas transações usando o pattern "Command". Ou seja, cada transação é representada por um objeto próprio. As aplicações de demo incluídas na distribuição do Prevayler exemplificam isso bem.

Com todos eles na RAM, serei capaz de usar ferramentas SQL para pesquisar os atributos de meus objetos?
Você pode até usar uma "SQL engine" qualquer que funcione em RAM. Eu não aconselharia isso, porém. A boa notícia é que você não vai mais estar quebrando descaradamente o encapsulamento de seus objetos.

Como é, então, que aplicativos cliente e outros sistemas vão acessar meus objetos?
RMI, CORBA, XML, SOAP: é você quem sabe. É exatamente para isso que servem todos esses padrões de middleware.

Como ficam as aplicações Web?

O Prevayler torna finalmente possível usar JSPs ou servlets acessando os objetos de negócios direto na mesma VM, ou seja, uma arquitetura que é, ao mesmo tempo, ridiculamente simples e absurdamente rápida.

O Prevayler é confiável?

A robustez do Prevayler está em sua simplicidade. Ele é ordens de grandeza mais simples que o mais simples banco de dados relacional. Além disso, sua licença de código aberto permite a toda a comunidade de desenvolvimento examinar, otimizar e ampliar o Prevayler. O que você também deve se perguntar é quão robusto é o seu próprio código. Lembre-se: você não estará mais escrevendo codigozinho apenas para clientes de um servidor de dados. Você tem finalmente a possibilidade de escrever o código de um verdadeiro servidor de objetos. Esse é o intuito da orientação a objetos desde o início, mas certamente não é para qualquer um.

Quem já está usando o Prevayler?

Tem como eu rodar o teste de escalabilidade?

O Prevayler já foi portado para outras linguagens?

Tem algum sisteminha de demo?

As respostas para essas e muitas outras perguntas estão no site do Prevayler: www.prevayler.org.

Mas e se eu estiver emocionalmente ligado à minha base de dados?

Para muitas aplicações, a prevalência é uma forma muito mais rápida, muito mais barata e muito mais simples de persistir seus objetos para futuras gerações. É claro que vão inventar todo tipo de desculpa para ficar pendurados ao “bom e velho banco de dados”, mas, ao menos agora, você tem alternativa.