Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

v. 11, n. 1
jan-jun 2012

doi:10.5329/RESI.2012.1101

Sumário

ISSN: 1677-3071

Revista hospedada em: http://revistas.facecla.com.br/index.php/reinfo
Forma de avaliação: *double blind review*

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente, mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.

PROPAGAÇÃO DE IDENTIDADE E EXECUÇÃO DE REGRAS DE AUTORIZAÇÃO PARA CONTROLE DE ACESSO EFETIVO EM SISTEMAS DE INFORMAÇÃO

PROPAGATION OF IDENTITY AND EXECUTION OF AUTHORIZATION RULES FOR EFFECTIVE INFORMATION SYSTEMS' DATA ACCESS CONTROL

(artigo submetido em agosto de 2011)

Felipe Leão
Aluno do Bacharelado em SI
Depto. de Informática Aplicada - Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
felipe.leao@uniriotec.br

Sérgio Gonçalves Puntar
Aluno do Programa de Pós-Graduação em Informática - Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
sergio.puntar@uniriotec.br

Leonardo Guerreiro Azevedo
Depto. de Informática Aplicada -Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
azevedo@uniriotec.br

Fernanda Baião
Depto. de Informática Aplicada -Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
fernanda.baiao@uniriotec.br

Cláudia Capelli
Depto. de Informática Aplicada -Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
claudia.cappelli@uniriotec.br

ABSTRACT

Information security is a critical issue for organizations and comprises several perspectives, including information systems development and data repositories accessed by those systems. In current distributed information systems architectures, information security involves identity propagation and data access authorization issues. This work describes the design and implementation of a simple, yet not trivial, architecture for effectively assuring information security, based on a series of standard technologies.

Key-words: database security; identity propagation; dependency injection; data access control; authorization rules; TPC-H benchmark.

RESUMO

Segurança de informação é uma questão crítica em organizações que deve ser tratada sob diferentes perspectivas, incluindo a do desenvolvimento dos sistemas de informação e dos repositórios dos dados acessados por tais sistemas. Nos diversos cenários de arquitetura distribuída dos sistemas de informação atuais, o problema da segurança da informação envolve as questões de propagação de identidade e de regras de autorização de acesso aos dados. Este artigo descreve o projeto e implementação de uma arquitetura eficaz para solução destes problemas que integra, de forma simples e não trivial, uma série de tecnologias disponíveis.

Palavras-chave: segurança em banco de dados; propagação de identidade; injeção de dependência; controle de acesso a dados; regras de autorização; benchmark TPC-H.

1 INTRODUÇÃO

Segurança é um dos principais temas discutidos atualmente nas organizações. Dentro dele, aspectos de controle de acesso, controle de interface, auditoria, controle do fluxo de informação, disponibilidade e confidencialidade são tratados. Neste universo, o aspecto de controle (ou autorização) de acesso, responsável em grande parte pela garantia da integridade se apresenta como elemento central (SANDHU *et al.*, 1996; YANG, 2009; CALÌ e MARTINENGHI, 2008; MURTHY e SEDLAR, 2007). É por meio de mecanismos que implementam este aspecto, por exemplo, que regras de negócio são garantidas. Regras de negócio são declarações que definem ou restringem algum aspecto de uma organização (BRG, 2000). Elas têm como objetivo definir a estrutura de um negócio ou controlar ou influenciar o seu comportamento. Em particular, uma categoria de regras de negócio é a assertiva de ação de autorização, ou **regra de autorização**, a qual restringe a **quem** é permitido realizar uma **ação** na organização e sobre quais **dados**.

Em diversos cenários de arquitetura distribuída dos sistemas de informação atuais, onde as camadas de um sistema (aplicação cliente, servidor de aplicação, servidor de dados) encontram-se implementadas em componentes independentes e alocadas a unidades de processamento distintas, os principais mecanismos para implementação do controle de acesso são a propagação de identidade entre as camadas do sistema de informação e a aplicação de regras de autorização. Estes mecanismos fazem com que a identidade do usuário seja propagada até o mecanismo de acesso a dados (por exemplo, um SGBD - Sistema de Gerenciamento de Banco de Dados), de modo que sejam manipulados apenas os dados a que o usuário tem acesso e também que sejam executadas sobre estes dados apenas as operações que o usuário pode fazer com o dado acessado.

Este artigo descreve o projeto de uma arquitetura eficaz para implementação de mecanismos de propagação de identidade e execução de regras de autorização, integrando de forma simples, porém não trivial, uma série de tecnologias disponíveis no mercado. Este artigo corresponde a uma extensão do trabalho de Leão *et al.* (2011). O protótipo foi testado quanto a sua eficácia em um ambiente de simulação, empregando o benchmark TPC-H, tendo obtido resultados satisfatórios.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta quatro cenários de aplicações para acesso a banco de dados, considerados pela solução proposta. A Seção 3 define mecanismos para autorização de acesso a banco de dados. A Seção 4 apresenta a proposta de arquitetura para controle de acesso por meio de propagação de identidade. A Seção 5 apresenta o projeto e a implementação da proposta, enquanto que a Seção 6 apresenta os testes experimentais realizados. Finalmente, a Seção 7 conclui o trabalho.

2 CENÁRIOS DE ARQUITETURA DISTRIBUÍDA DE SISTEMAS DE INFORMAÇÃO

Esta seção descreve quatro cenários genéricos de arquiteturas de sistemas de informação em ambientes distribuídos que são freqüentemente encontrados nas organizações.

2.1 CLIENTE ACESSA O SGBD DIRETAMENTE EXECUTANDO UM PROCESSO IDENTIFICÁVEL

Este cenário contempla o conjunto de aplicações que, por meio de processos executados no computador do cliente, fazem acesso direto ao SGBD para acesso ao banco de dados. Estes processos são identificáveis porque são executados a partir do usuário real da aplicação, que é o usuário do sistema operacional. Este cenário é ilustrado na Figura 1, onde o usuário *C123* utilizando a aplicação cliente é o mesmo usuário reconhecido pelo SGBD. O SGBD é acessado po uma conexão direta, por exemplo, conexão JDBC[1], que utiliza um usuário de banco de dados único para cada aplicação. Entretanto, o SGBD é capaz de capturar parâmetros do ambiente de conexão, como, por exemplo, o usuário do sistema operacional que corresponde ao usuário real, a partir de variáveis do SGBD de gerencia ao acesso ao banco de dados. Um exemplo é o atributo OS_USER existente no *namespace userenv*[2] do SGBD Oracle. Este cenário contempla aplicações cliente-servidor, como, por exemplo, aplicações clientes do SGBD como o SQL Plus[3].

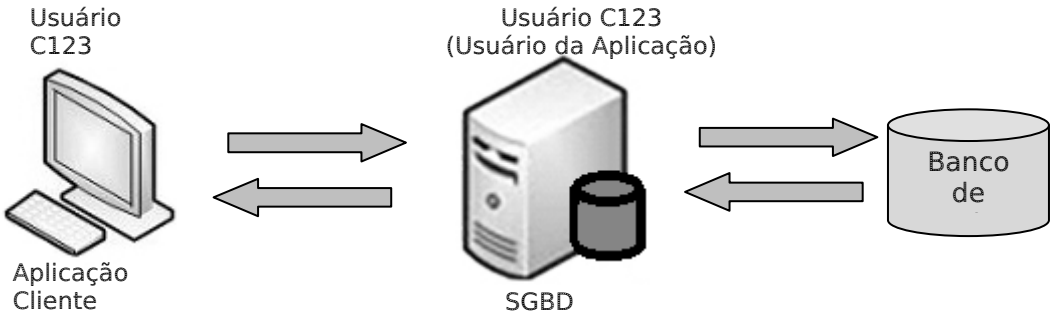

Figura 1. Cenário 1: Cliente acessa o SGBD diretamente executando um processo identificável
Fonte: elaborada pelos autores

2.2 CLIENTE ACESSA O SGBD DIRETAMENTE EXECUTANDO UM PROCESSO NÃO IDENTIFICÁVEL

A diferença do segundo cenário tecnológico em relação ao primeiro está em como o processo[4] que consome e processa dados do banco de dados é executado. Este processo executado no cliente é iniciado com um

[1] http://java.sun.com/products/jdbc/overview.html
[2] Userenv é um namespace do SGBD Oracle para acesso a informações da sessão de um cliente.
[3] http://www.oracle.com/technology/docs/tech/sql_plus/index.html
[4] Entende-se por processo um executável ou serviço rodando no sistema operacional que acessa o banco de dados.

usuário do processo distinto do usuário do sistema operacional. Por exemplo, um processo *P* executando no sistema operacional que acessa o SGBD é executado por um usuário privilegiado do sistema operacional chamado *UsrP* ao invés do usuário logado *UsrL*. Este cenário é ilustrado na Figura 2, onde o usuário logado no sistema operacional é o usuário *C123*, mas o usuário utilizado para executar a aplicação cliente é o usuário *APL1*, que é o usuário reconhecido pelo SGBD como o usuário executando o processo. Portanto, não é trivial a identificação do usuário da aplicação a partir da conexão com o SGBD advinda destes processos. Na abordagem anterior, o parâmetro OS_USER traria o usuário genérico da aplicação para acesso ao SGBD. Isso impossibilitaria, o processo de autorização dos dados por desconhecer quem é o usuário real da aplicação.

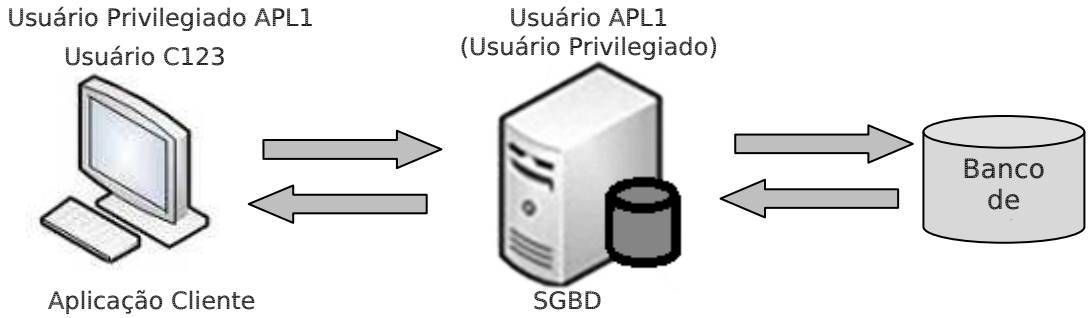

Figura 2. Cenário 2: Cliente acessa o SGBD diretamente executando um processo não identificável
Fonte: elaborada pelos autores

2.3 CLIENTE ACESSA SERVIDOR DE APLICAÇÃO QUE ACESSA O SGBD

No terceiro cenário, está o conjunto de aplicações que possuem um cliente que deve se conectar a um servidor de aplicação para executar um objeto remoto que, por sua vez, acessa o SGBD. Neste cenário, o servidor de aplicação acessa o SGBD por meio de um usuário genérico do banco de dados, não sendo uma tarefa simples identificar o usuário que executou a aplicação cliente. Como exemplificado na Figura 3, o usuário identificado pelo SGBD é o usuário utilizado pelo servidor de aplicação para se conectar ao SGBD (neste caso, *SRV1*), o qual não é o usuário utilizado para acessar a aplicação cliente (neste caso, *C123*).

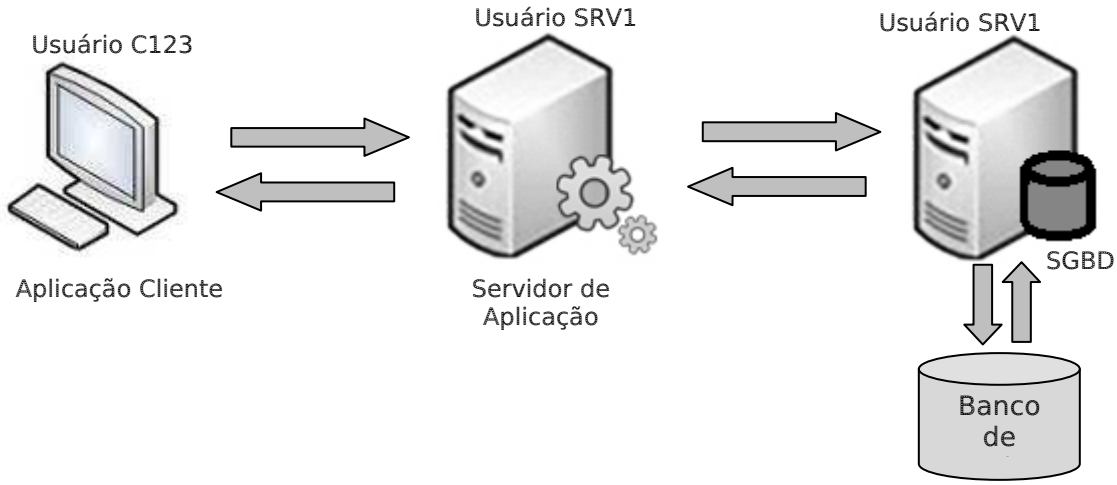

Figura 3. Cenário 3: Cliente acessa servidor de aplicação que acessa o SGBD
Fonte: elaborada pelos autores

2.4 CLIENTE ACESSA SGBD VIA SERVIÇOS WEB

No quarto cenário, a arquitetura da aplicação é composta por um servidor de aplicação que acessa o SGBD executando serviços Web usando um ou mais usuários criados por instância do SGBD.

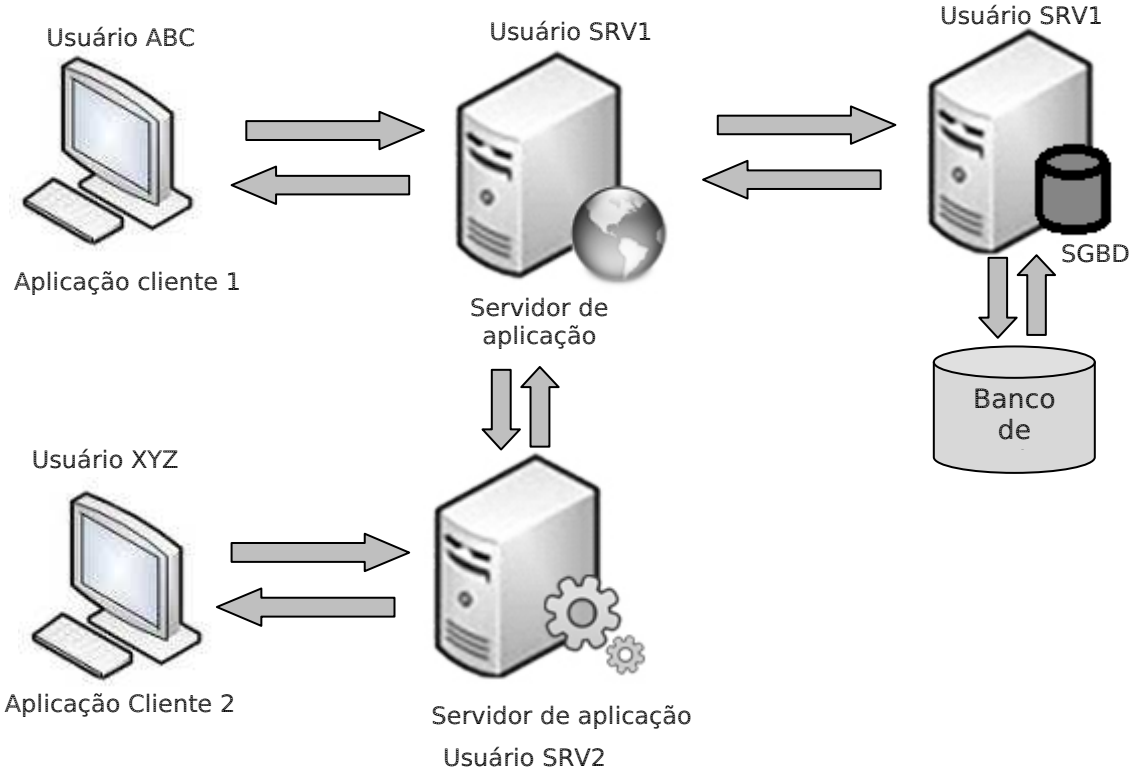

Figura 4. Cenário 4: Cliente acessa SGBD via serviços web
Fonte: elaborada pelos autores

Neste cenário, os clientes que acessam este servidor de aplicação podem ser outras aplicações, o que deixaria o usuário real em uma camada ainda mais distante. Este cenário é ilustrado na Figura 4, onde temos dois exemplos de conexão utilizando serviço. Em um dos exemplos, o usuário *ABC* está utilizando a aplicação cliente 1 que invoca um serviço web. O serviço web, por sua vez utiliza o usuário *SRV1* para se conectar ao SGBD, o qual é reconhecido pelo SGBD, ao invés do usuário *ABC*. Já no outro exemplo, o usuário *XYZ* está executando a aplicação cliente 2 que invoca um componente remoto no servidor de aplicação, o qual não é executado com o usuário *XYZ*. Este componente, por sua vez, invoca o serviço web que utiliza o usuário *SRV1* para se conectar ao SGBD. Novamente, o usuário reconhecido pelo SGBD é o usuário *SRV1* e não o usuário que está utilizando a aplicação cliente.

3 MECANISMOS PARA CONTROLE DE AUTORIZAÇÃO DE ACESSO A DADOS

Existem diferentes mecanismos de controle de autorização de acesso a dados, os quais podem ser divididos em DAC (*Discricionary Access Control*), MAC (*Mandatory Access Control*), ambos propostos por (DoD, 1983) e RBAC (*Role-Based Access Control*) (FERRAIOLO e KHUN, 1992).

O mecanismo DAC restringe acesso a objetos baseado na identidade de usuários e/ou grupos aos quais eles pertencem. Yang (2009) aponta que políticas DAC não garantem controle sobre o fluxo de dados, pois tornam possível que dados cheguem a usuários que não têm permissão de lê-las. Políticas MAC, também conhecidas como *label security*, são baseadas em regulamentações mandatórias determinadas por uma autoridade central (YANG, 2009). A forma mais comum de MAC é a política de segurança de múltiplos níveis usando classificação de sujeitos (usuários ou grupos) e objetos (dados) dos sistemas. MAC controla o fluxo de dados, embora não aborde as ações que podem ser executadas pelos sujeitos sobre os dados (FERRAIOLO e KHUN, 1992). Além disso, como um rótulo é atribuído a cada instância de dados, o custo de gestão e de manutenção dos rótulos é alto, e pouco flexível, se existirem muitas políticas definidas. No caso de RBAC, o controle de acesso considera funções e operações, além de dados. Neste caso, o interesse principal é proteger a integridade do dado, definindo quem pode realizar qual ação sobre qual dado (FERRAIOLO e KHUN, 1992). Ferraiolo *et al*. (2001) propõem um padrão para RBAC a fim de unificar ideias existentes em diferentes modelos de referência de RBAC, produtos comerciais e protótipos de pesquisa.

Os mecanismos para controle de acesso encontram-se implementados em diferentes Sistemas de Gerenciamento de Banco de Dados (SGBD), como Oracle, Sybase, Microsoft SQL-Server (YANG, 2009). Portanto, o acesso dos usuários utilizando as aplicações disponibilizadas pelas organizações poderia ser controlado diretamente pelos mecanismos de controle de acesso existentes nos SGBDs. No entanto, existem diferentes cenários de conexão das aplicações com os SGBDs, como detalhado na Seção 2. Para

alguns cenários o uso das funcionalidades dos SGBDs é simples, sendo fácil identificar o usuário que está logado na aplicação e, a partir daí, aplicar as regras que se referem a ele. Por outro lado, em outros cenários não é simples identificar o usuário diretamente, sendo necessário utilizar outros mecanismos para que o SGBD consiga aplicar as regras de autorização sobre o usuário que está utilizando a aplicação.

4 UMA ARQUITETURA PRÁTICA E EFICAZ PARA O ASPECTO DE CONTROLE DE ACESSO

Nos cenários de arquitetura de Sistemas de Informação descritos na Seção 2, a solução para o aspecto de controle de acesso passa por implementar mecanismos relacionados à definição dos perfis e das regras de autorização; à autenticação do usuário; à propagação de identidade do usuário autenticado entre as camadas do sistema até o SGBD e, finalmente, uma vez que o usuário atual do sistema de informação tenha sido reconhecido pelo SGBD, à execução das regras de autorização em tempo de execução. Esta Seção apresenta a arquitetura para controle de acesso proposta, descrevendo em detalhes os aspectos de execução das regras de autorização (Seção 4.1) e de propagação de identidade (Seção 4.2). A implementação de mecanismos relacionados à definição de perfis e regras de autorização foi tratada por Azevedo *et al.* (2010) e é apresentada na Seção 4.1. A implementação de mecanismos de autenticação do usuário está fora do escopo deste trabalho.

4.1 MECANISMOS PARA EXECUÇÃO DE REGRAS DE AUTORIZAÇÃO DE ACESSO A DADOS

A solução da arquitetura proposta neste trabalho para execução de regras de autorização de acesso a dados considera o uso do mecanismo de controle de acesso RBAC implementado no SGBD Oracle empregando a proposta de modelo flexível para controle de acesso proposta por Azevedo *et al.* 2011, denominada FARBAC (*Flexible Approach for Role-Based Access Control*). A avaliação deste modelo foi implementada utilizando o mecanismo VPD (*Virtual Private Database*) (JELOKA *et al.*, 2008) presente no SGBD Oracle.

No FARBAC, as regras de autorização são definidas como cláusulas WHERE, que são dinamicamente adicionadas aos comandos enviados pelas aplicações ao SGBD. Em seleções, por exemplo, as regras funcionam como um filtro, removendo da consulta os registros a que o usuário não possui acesso. As regras são associadas a perfis (ou papéis) que são concedidos a cada usuário. O FARBAC recupera em tempo de execução o usuário, executando o comando, e aplica as regras referentes aos perfis desse usuário.

Uma premissa para o funcionamento do FARBAC é que a identidade do usuário seja propagada corretamente nas diversas camadas do sistema. Na atual implementação do modelo no Oracle, a identidade do usuário chega ao SGBD por meio da chamada ao procedimento *dbms_application_*

info.set_client_info, o qual define um valor para o atributo *client_info* do *namespace userenv*, posteriormente recuperado pelo FARBAC.

4.1.1 MODELO CONCEITUAL DAS REGRAS DE AUTORIZAÇÃO

As regras de autorização do FARBAC são armazenadas no banco de dados de regras de negócio, de acordo com o modelo conceitual de entidades e relacionamentos apresentado na Figura 5 e descrito a seguir.

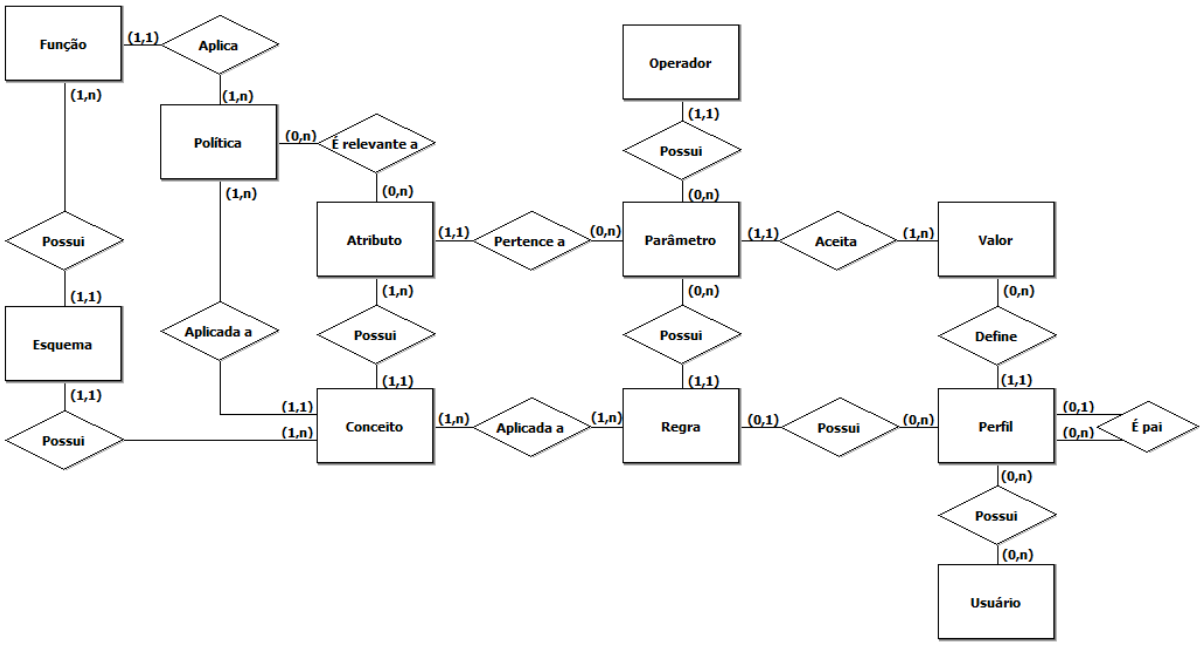

Figura 5. Modelo de entidade e relacionamento do FARBAC
Fonte: Leão *et al.* (2011)

Os **Usuários** são agrupados em **Perfis**, sendo que cada usuário pode ser associado a diversos perfis e cada perfil pode agrupar diversos usuários. O auto-relacionamento da entidade **Perfil** permite a representação de uma hierarquia de perfis. Por exemplo, os perfis de Gerente de Vendas da América e Ásia e de Gerente de Vendas da Europa podem ser especializações do perfil Gerente de Vendas. Isso significa que regras de autorização atribuídas a um perfil mais geral (Gerente de Vendas) devem também ser aplicadas a todos os seus descendentes. Além disso, perfis descendentes podem determinar valores específicos para os parâmetros do predicado. Por exemplo, o perfil Gerente de Vendas pode restringir o acesso aos pedidos baseado no atributo continenteDeOrigem do conceito Pedido; o perfil Gerente de Vendas da América e Ásia pode então especificar o(s) valor(es) para o atributo controlado (por exemplo, “America, Asia”). Dessa forma é possível que a política RBAC implemente a regra, restringindo o acesso do perfil Gerente de Vendas América e Ásia somente para os pedidos onde “*continenteDeOrigem in (‘America’,‘Asia’)*”.

A entidade **Regra** representa as definições de regras que são armazenadas como cláusulas SQL, que serão usadas para transformar os comandos SQL provenientes da aplicação cliente. Uma cláusula restringe o acesso do **Usuário** a um subconjunto dos dados de um **Conceito**, que pode

ser uma tabela, uma visão ou um sinônimo. Portanto, existe um relacionamento **Regra** × **Conceito**. A mesma cláusula SQL pode ser usada para definir regras de autorização em diferentes conceitos e cada **Conceito** pode possuir mais de uma **Regra**.

Um **Parâmetro** denota uma expressão no formato "Atributo Operador Valor" relacionando as entidades correspondentes (**Atributo** × **Operador** × **Valor**). A entidade **Atributo** representa um atributo de um conceito do banco de dados utilizado para restrição de acesso. A entidade **Operador** representa um operador binário (incluindo =, <>, >, >=, <, <=, IN e NOT IN). A entidade **Valor** representa um valor (ou lista de valores) dentro do domínio do atributo, que delimita um subconjunto de dados. O relacionamento **Conceito** × **Atributo** indica a que conceito um atributo pertence, da mesma forma que o relacionamento **Esquema** × **Conceito** indica a que esquema o conceito pertence. Esse relacionamento denota uma representação em alto nível do esquema do banco de dados para o qual as regras de autorização são aplicadas.

A **Política** dita o controle de acesso que deve ser aplicado a um conceito. Uma política de controle de acesso é aplicada por uma **Função** (que também pertence a um **Esquema**). Durante sua execução, esta função identifica o *Conceito* sendo acessado e o *Usuário* executando o comando. Ela captura os dados dos outros elementos do modelo para construir o predicado da regra de autorização (ou predicado de autorização) daquele conceito para aquele usuário. Além disso, o relacionamento **Política** × **Atributo** determina quais atributos do conceito possuem dados sensíveis.

O mesmo usuário pode possuir mais de um perfil relacionado a um mesmo conceito. Dessa forma, em tempo de execução, mais de uma regra (predicado de autorização) será retornada, exigindo que as regras sejam compostas de alguma forma. Essa composição pode ser feita por meio do uso dos operadores OR ou AND. A composição por OR, que chamamos de composição permissiva, permite que o usuário tenha acesso à união dos subconjuntos de dados permitidos por cada regra. A composição por AND, que chamamos de composição restritiva, limita o acesso do usuário à interseção dos subconjuntos de dados permitidos pelas regras combinadas. Por exemplo, um usuário que possua os perfis Gerente de Vendas da Europa e Gerente de Vendas da França, com a composição permissiva teria acesso aos dados de toda a Europa, já com a composição restritiva o seu acesso seria apenas aos dados da França. Tanto a composição permissiva quanto a composição restritiva são suportadas pelo FARBAC e é responsabilidade da organização decidir que tipo de composição usar.

4.1.2 ALGORITMO PARA CONSTRUÇÃO DO PREDICADO DE AUTORIZAÇÃO

O FARBAC possui uma função genérica responsável por construir, em tempo de execução, o predicado de autorização para cada conceito. O predicado de autorização realiza a composição (permissiva ou restritiva) de todas as regras de um usuário para um conceito. A função é aplicada a

todos os conceitos com dados sensíveis e retorna o predicado de autorização, dependendo do usuário que realiza o acesso e dos dados cadastrados no modelo de armazenamento de regras de autorização (Figura 6).

```
Recupera o usuário que está executando a consulta
A partir do relacionamento Usuário × Perfil, recupera os perfis do usuário
Para cada perfil Faça
    A partir do relacionamento Perfil × Regra × Conceito, identifica as regras
    que devem ser aplicadas de acordo com o perfil pai da hierarquia
    Para cada regra Faça
        A partir do relacionamento Regra × Parâmetro, identifica os parâmetros
        específicos da regra
        Para cada parâmetro Faça
            A partir do relacionamento Perfil × Valor, recuperar o(s) valor(s)
            aceito(s) pelo parâmetro para filtragem dos dados
Finalmente, o predicado de autorização é montado e retornado
```

Figura 6. Algoritmo de construção do predicado de autorização
Fonte: Leão *et al.* (2011)

4.2 MECANISMOS PARA PROPAGAÇÃO DE IDENTIDADE

A propagação de identidade procura permitir que a identidade do usuário seja preservada, independentemente de onde a informação da identidade foi criada, para utilização durante autorização e para fins de auditoria (IBM, 2010). A proposta para propagação de identidade é utilizar o padrão de injeção de dependência (do inglês *Dependency Injection*), o qual é um padrão de projeto derivado do padrão Inversão de Controle. Assim como seu genitor, visa a fornecer a uma classe informações necessárias para o seu funcionamento sem que a classe tenha que se preocupar com a geração destes dados. Neste caso, propomos o uso de injeção via *setter* (*Setter Injection*), a qual consiste em passar as dependências por meio de argumentos por um método *setter* (PRASANNA, 2009).

4.3 ARQUITETURA PARA PROPAGAÇÃO DE IDENTIDADE E AUTORIZAÇÃO DE ACESSO

A proposta de arquitetura para propagação de identidade e execução de regras de autorização é ilustrada na Figura 7. É importante enfatizar que a proposta considera que o usuário do sistema de informação está devidamente autenticado para a propagação de sua identidade. Segundo Needham e Schroeder (1978), autenticação corresponde à verificação mútua da identidade das partes que estão se comunicando.

Tendo em mãos as informações do usuário autenticado, a aplicação cliente repassa estas informações para as camadas seguintes até chegar ao SGBD. O SGBD, por sua vez, implementa a arquitetura do FARBAC (descrita na Seção 4.1). Na Figura 7, é feita a relação entre os cenários apresentados na Seção 2 e a proposta: (i) cenários 1 e 2 - aplicação acessando o SGBD diretamente: as informações do usuário são passadas diretamente da aplicação para o SGBD para realizar acesso controlado ao banco de dados; (ii) cenário 3: aplicação cliente passa as informações do usuário para o servidor de aplicação que as passa para o SGBD; (iii) cenário 4 – múltiplas camadas: aplicação cliente passa as informações do

usuário para um servidor de aplicação, no qual existem serviços web executando, e este repassa ao SGBD. Considerando ainda o cenário 4, também é possível considerar que a aplicação cliente acessa o servidor de aplicação (ii) que acessa o servidor de serviços web que acessa o SGBD. Neste caso, as informações devem ser passadas pela aplicação cliente para o servidor de aplicação, deste para o serviço web e, em seguida, este repassa para o SGBD.

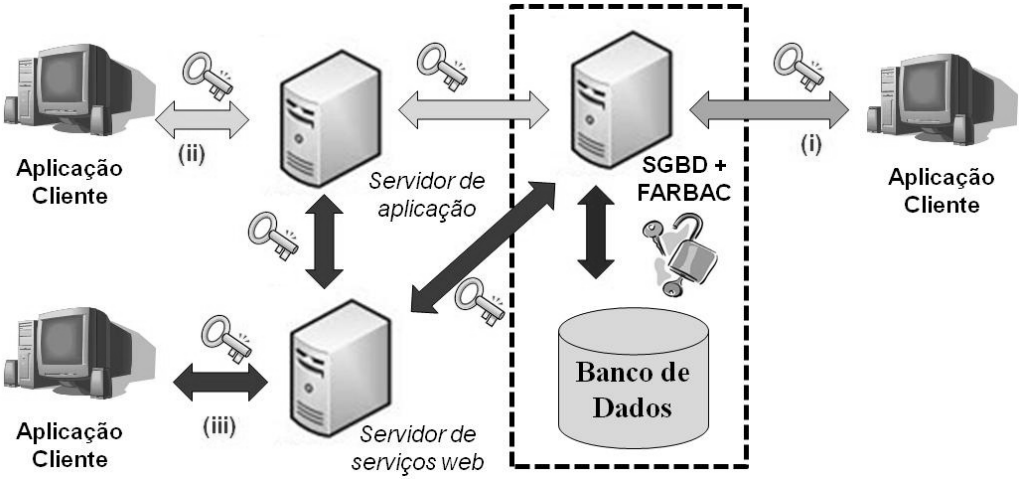

Figura 7. Propagação de identidade nos cenários da arquitetura proposta
Fonte: Leão *et al.* (2011)

5 PROJETO E IMPLEMENTAÇÃO DA ARQUITETURA PROPOSTA

A implementação de mecanismos para propagação de identidade do usuário segundo a arquitetura proposta contempla três dos quatro cenários descritos na Seção 2 e emprega tecnologias de amplo uso no mercado e padrões de programação.

Um protótipo foi implementado utilizando as tecnologias Java Enterprise Edition e Standart Edition, Enterprise Java Beans 3.0 (EJB3), servidor de aplicação JBoss (versão 4.2), *framework* de mapeamento objeto-relacional Hibernate (versão 3), SGBD Oracle (versão 10g) e os padrões de projeto *Service Locator*, *Dependency Injection* e *Data Access Object* (DAO). O código está disponibilizado em Googlecode[5]. O banco de dados utilizado para os testes seguiu o esquema do *benchmark* TPC-H (TPC COUNCIL, 2008), o qual consiste em uma especificação de grande relevância na indústria que simula um cenário de uma aplicação de apoio à decisão. O protótipo implementado trata os primeiros três cenários apresentados na Seção 2. Para o quarto cenário, está sendo avaliado o uso de padrões de web services, tais como, SOAP[6], WS-Security[7] e WS-Policy[8] para transportar de forma segura as informações do usuário no cabeçalho da mensagem

[5] http://pdac.googlecode.com/
[6] http://www.w3.org/TR/soap/
[7] http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
[8] http://www.w3.org/TR/ws-policy/

SOAP. No entanto, vale ressaltar que os aspectos de propagação de identidade apresentados neste trabalho são os mesmos que estão sendo avaliados para o uso destas tecnologias.

O protótipo (Figura 8) consiste em uma aplicação cliente (Java SE), que acessa um EJB *Stateful* responsável por realizar operações de consulta ao banco de dados Oracle e devolver o resultado da pesquisa à aplicação cliente. O acesso ao EJB é realizado por meio do padrão de projeto *Service Locator*. O EJB acessa o SGBD por meio do *framework* de mapeamento objeto-relacional Hibernate.

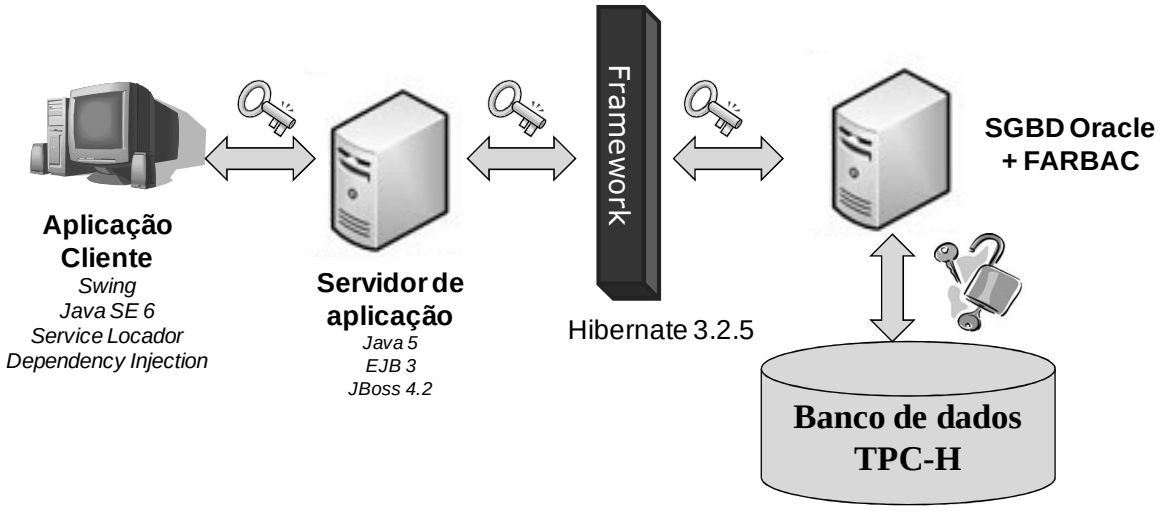

Figura 8. Arquitetura do protótipo
Fonte: Leão *et al.* (2011)

5.1 APLICAÇÃO CLIENTE

Na arquitetura do protótipo implementado, a aplicação cliente utiliza a tecnologia Java *Swing* para solicitar a execução de uma consulta no banco de dados. O passo-a-passo de execução de uma requisição na aplicação Cliente é apresentado a seguir e ilustrado na Figura 9. Para auxiliar na compreensão, o fluxo de execução do módulo servidor foi abstraído em uma única raia (EJB) que será explicada detalhadamente na seção 0 deste trabalho.

1. No módulo cliente, após o *login* e autenticação do usuário (Figura 9 - passo 1), a instância única do ServiceLocator é recuperada e nela são armazenadas as informações do usuário (contendo, por exemplo, a chave do usuário) pela invocação do método setUsuario do ServiceLocator (Figura 9 - passo 1.1). Como o ServiceLocator é um Singleton, a injeção do usuário pode ser realizada uma única vez, preferencialmente após a autenticação do usuário.
2. O usuário acessa uma funcionalidade da aplicação, por exemplo, a consulta de pedidos. Neste momento, o método consultarPedido de PedidosBO é invocado (Figura 9 - passo 2).
3. O Business Object PedidosBO então requisita ao ServiceLocator o

acesso ao componente remoto (um componente EJB) que executará a consulta no banco de dados propriamente dito. A busca pelo objeto remoto é feita por meio do nome JNDI deste EJB. Neste caso, PedidosBO invoca o método get do ServiceLocator (**Figura 9** - passo 2.1).

4. O ServiceLocator então recupera o contexto e invoca o método *lookup* neste contexto para buscar por uma referência ao EJB remoto (**Figura 9** - passo 2.1.1).
5. Ao receber a referência do componente EJB, o ServiceLocator verifica se o EJB recuperado implementa a interface ControleAcesso e, em caso positivo, invoca o método "setUsuario" (**Figura 9** – passo 2.1.2), passando as informações do usuário (com a chave do mesmo, por exemplo) para armazenar as informações do usuário no EJB. A referência ao EJB então é devolvido para o BO PedidosBO.
6. O BO PedidosBO invoca o método do EJB remoto que irá realizar a consulta do desconto no banco de dados (**Figura 9** – passo 2.2).
7. O EJB executa a consulta e retorna os dados para o BO PedidosBO (**Figura 9** – "Valor Final", à direita).
8. O BO PedidosBO retorna os dados para a aplicação poder apresentá-las para o usuário (**Figura 9** – "Valor Final", à esquerda).

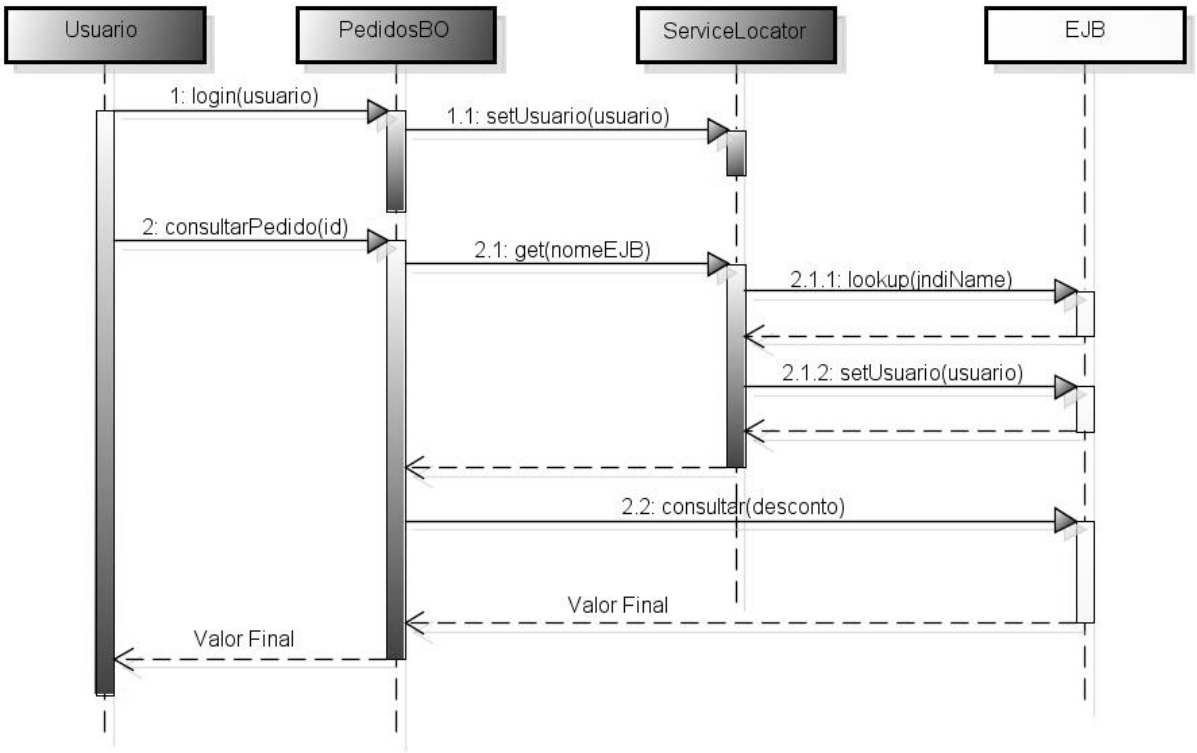

Figura 9. Diagrama de sequência da aplicação cliente
Fonte: elaborada pelos autores

5.2 SERVIDOR DE APLICAÇÃO

No servidor de aplicação, as regras de negócio estão implementadas em projetos Java utilizando a tecnologia EJB (*Stateful*). Neste caso, temos uma classe Java EJB para cada projeto EJB. O acesso e persistência de dados foram implementados utilizando o *framework* de mapeamento objeto relacional Hibernate[9], acessando SGBD Oracle. A responsabilidade de gerenciar o ciclo de vida das conexões com o banco de dados (*pool* de conexão) e o contexto transacional é do servidor de aplicação JBoss[10].

Uma interface, chamada *ControleAcesso*, foi implementada para especificar os métodos que os EJB devem implementar para armazenamento das informações necessárias para controle de acesso aos dados. Esta interface é apresentada na Figura 10.

```
1  public interface ControleAcesso {
2      void setUsuario(String chaveUsuario);
3      String getUsuario();
4  }
```

Figura 10. Implementação da Interface *ControleAcesso*
Fonte: Leão *et al.* (2011)

A interface *ControleAcesso* é parte de uma biblioteca independente, devendo ser importada para o *classpath* de cada projeto EJB antes de ser utilizada. Esta separação permite que, posteriormente, outros projetos possam utilizar a mesma interface ao implementar controle de acesso aos dados.

Em cada projeto EJB, é criada uma interface com o objetivo de expor os métodos que poderão ser acessados pelo cliente. Esta interface deve herdar a interface *ControleAcesso*, obrigando o EJB a implementar também os métodos *setUsuario* e *getUsuario* de *ControleAcesso*. O método *setUsuario* é utilizado pela aplicação cliente para ajustar no EJB as informações do usuário, de acordo com o padrão de projeto Injeção de Dependência, cujo uso é proposto na arquitetura.

O acesso ao banco de dados foi implementado utilizando o padrão *HibernateUtil*[11] sugerido pelos desenvolvedores do *framework* Hibernate, centralizando a inicialização do Hibernate e o *lookup* à fábrica de sessões com o banco de dados. Além disso, foi realizada uma modificação nesta classe, onde, ao invés de a aplicação requisitar ao *HibernateUtil* uma fábrica de sessões, ela deverá requisitar diretamente uma sessão. O objetivo é informar ao SGBD o usuário que irá realizar as operações no banco de dados. A Figura 11 apresenta a implementação do método *getSession*, da classe *HibernateUtil*, em que é criada uma sessão com o banco de dados. O método armazena as informações do usuário no SGBD antes de retornar o objeto de sessão para quem invocou o método. Neste caso,

[9] http://www.hibernate.org/
[10] O servidor de aplicação utilizado é o JBoss 4.2
[11] A implementação base para a classe HibernateUtil é demonstrada na documentação do Hibernate e pode ser vista em http://community.jboss.org/wiki/sessionsandtransactions.

como o SGBD utilizado é o Oracle, o armazenamento das informações do usuário é feito por meio da chamada ao procedimento *dbms_application_info.set_client_info*. Esta chamada armazena a chave do usuário no atributo *client_info* do *namespace userenv* do contexto do banco de dados Oracle.

```
33      public Session getSession(String chaveUsuario) {
34          if (session.get() == null) {
35              session.set(sessionFactory.openSession());
36          }
37
38          //preparar o contexto da sessao no banco com a chave do usuario
39          Connection con = session.get().connection();
40          try{
41              CallableStatement st = con.prepareCall(
42                      "{call dbms_application_info.set_client_info(?)}");
43              st.setString(1, chaveUsuario);
44              st.executeUpdate();
45          }catch(SQLException e){
46              return null;
47          }
48
49          return session.get();
50      }
```

Figura 11. Método getSession da classe HibernateUtil
Fonte: Leão *et al.* (2011)

O passo-a-passo de atendimento a uma requisição do cliente por um EJB é apresentado a seguir e ilustrado na Figura 12. O diagrama apresentado na Figura 12 corresponde ao detalhamento do passo 2.2 mostrado na Figura 9:

1. Após as informações do usuário devidamente ajustadas no EJB remoto (Figura 9 - passo 2.1.2), a aplicação Cliente invoca o método *consulta(desconto)* do EJB remoto ManterPedidos. Na Figura 9, este passo corresponde ao passo 2.2, enquanto que na Figura 12 corresponde ao passo 1.
2. Em seguida, o EJB ManterPedidos invoca o DAO responsável por executar a consulta de desconto (PedidoDAO) (Figura 12 - passo 1.1).
3. O objeto DAO então recupera a instância única da classe HibernateUtil e invoca o método getSession informando a chave do usuário (Figura 12 – passo 1.1.1).
4. A classe HibernateUtil cria uma sessão com o banco de dados, por meio da chamada ao método buildSessionFactory (Figura 12 – passo 1.1.1.1) e openSession da classe Hibernate (Figura 12 – passo 1.1.1.2). Em seguida, ajusta o usuário que está fazendo uso da sessão (Figura 12 – passo 1.1.1.3), por meio da chamada à procedure *dbms_application_info.set_client_info*, e retorna o objeto de sessão para o DAO.
5. A classe DAO executa a consulta ao banco de dados utilizando a

sessão recuperada. Nesse momento, o FARBAC entra em ação para aplicar a regra de autorização cadastrada e retornar apenas os dados a que o usuário tem acesso. Em seguida, o DAO fecha a sessão (Figura 12 – passo 1.1.3).

6. O resultado da consulta é repassado para o EJB (Figura 12 – Resultado da Consulta, à esquerda) que repassa para a aplicação cliente (Figura 12 – Resultado da Consulta, à direita).

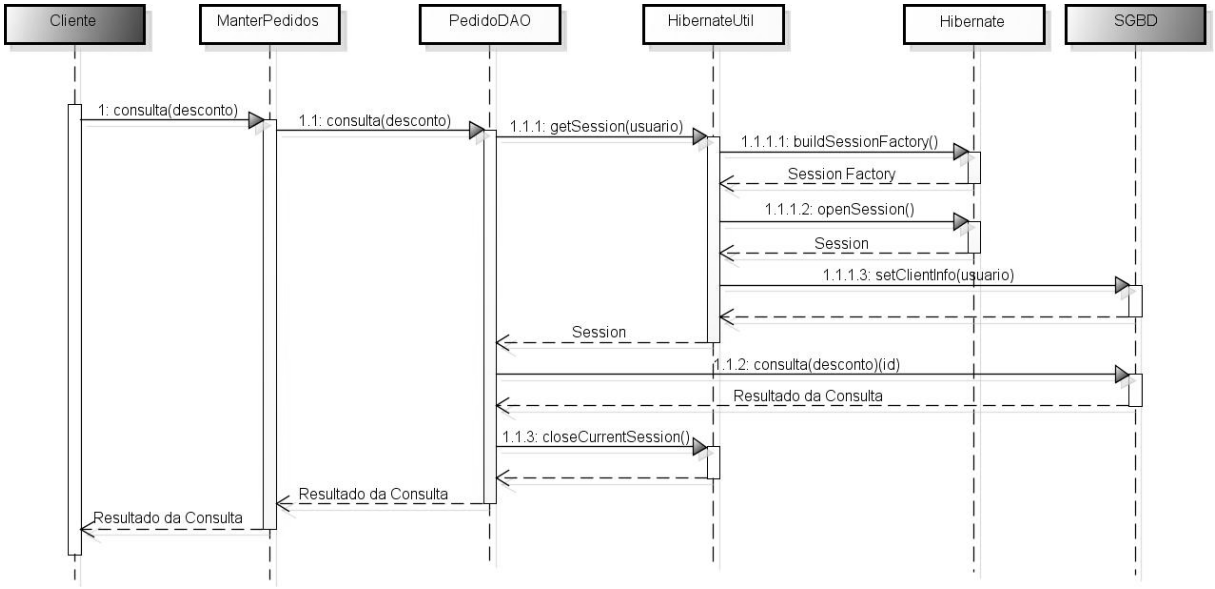

Figura 12. Diagrama de sequência para a aplicação servidor
Fonte: elaborada pelos autores

6 TESTES EXPERIMENTAIS

Para avaliar a arquitetura proposta, foi utilizado o esquema e os dados do *benchmark* TPC-H. O TPC-H é uma especificação de *benchmark* com uma ampla relevância na indústria, que simula o cenário de uma aplicação de apoio à decisão (TPC COUNCIL, 2008). Esse cenário possui um contexto realista que representa as atividades de uma indústria de atacado que precisa gerenciar as suas vendas ou distribuir um produto em todo o mundo (por exemplo, aluguel de veículos, distribuição de alimentos, etc.). O *benchmark* consiste da descrição de um esquema de banco de dados e de um conjunto de consultas *ad-hoc* OLAP (Online Analytical Processing) orientadas ao negócio. A consulta utilizada no teste foi a de previsão de mudança na receita, a qual quantifica o aumento da receita resultante da eliminação de algum desconto percentual em determinado ano (Figura 13).

```
select sum(l_extendedprice * l_discount) as revenue
from lineitem
where l_shipdate >= date '1994-01-01'
  and l_shipdate < date '1994-01-01' + interval '1' year
  and l_discount between 0.06 - 0.01 and 0.06 + 0.01
  and l_quantity < 24;
```

Figura 13. Previsão de mudança na receita
Fonte: Leão *et al.* (2011)

Foi definida a seguinte regra de autorização que foi cadastrada no modelo do FARBAC: “O gerente de vendas do norte da América e da Ásia deve ter acesso somente aos pedidos provenientes de fornecedores das nações do hemisfério Norte das regiões Ásia e América”. A implementação desta regra necessitou de algumas adaptações no modelo do TPC-H e no FARBAC, tais como: a chave do usuário na tabela *CUSTORMER* do TPCH referenciando à chave do usuário do FARBAC; inclusão de campo para nação do usuário no FARBAC; e inclusão de campo para indicar o hemisfério do usuário.

Para implementar essa regra de autorização foi definida uma hierarquia de perfis, com um perfil pai chamado *Gerente de Vendas* e um perfil filho chamado *Gerente de Vendas Norte América e Ásia*. Considerando a consulta, os dados a serem protegidos por essa política referem-se às tabelas *ORDERS* e *LINEITEM*, tabelas de pedidos e de itens de pedido, respectivamente. As informações para controle de acesso foram cadastradas no FARBAC e a função de autorização retorna os predicados referentes à Figura 14 e à Figura 15 para cada uma das tabelas. O predicado correspondente à tabela ORDERS (Figura 14) verifica se o cliente do pedido corresponde a um cliente que está na região América ou Ásia e no hemisfério norte. Já o predicado para a tabela LINEITEM faz acesso à tabela ORDERS. Ao fazer o acesso a esta tabela, a política correspondente é executada e, consequentemente, são retornados apenas os itens das regiões América e Ásia e do hemisfério norte.

```
(O_CUSTKEY in (
select C_CUSTKEY
from TPCH.CUSTOMER
inner join TPCH.NATION on N_NATIONKEY = C_NATIONKEY
inner join TPCH.REGION on R_REGIONKEY = N_REGIONKEY
where R_NAME IN ('AMERICA' , 'ASIA') AND N_HEMISPHERE IN ('NORTH')))
```

Figura 14. Predicado para a tabela ORDERS
Fonte: elaborada pelos autores

```
(L_ORDERKEY in (
select O_ORDERKEY
from TPCH.ORDERS))
```

Figura 15. Predicado para a tabela LINEITEM
Fonte: elaborada pelos autores

Foram criados os usuários: *Bob*, como gerente de vendas Norte da América e da Ásia; e *Alice*, que é a presidente da empresa e recebeu o privilégio *EXEMPT ACCESS POLICY*, que faz com que todas as políticas sejam ignoradas.

Em seguida, a Aplicação Cliente foi executada para invocar a consulta apresentada na Figura 13, com os usuário *Bob* e *Alice*. Como *Bob* é o gerente de vendas Norte da América e da Ásia, ele tem acesso apenas aos itens de pedidos do norte dessas regiões, enquanto *Alice*, que é a presidente, tem acesso a todos os itens de pedidos. Para *Alice*, a mudança na receita foi de R$ *88.894.386,60*. Já para *Bob* esta foi de apenas R$ *21.445.915,50*, afinal a sua visão é somente dos itens de pedidos do Norte da América e da Ásia.

CONCLUSÃO

Segurança é uma questão importante para as organizações. Em geral questões desta área são resolvidas com a implementação de mecanismos de controle de acesso e da implementação de regras de autorização em sistemas de informação. Contudo, quando uma regra muda, todos estes sistemas que implementam estas regras e controles devem ser atualizados, fazendo com que isso se torne um problema bastante complexo principalmente em cenários onde existem sistemas legados e um número muito grande de regras de autorização.

Grande parte das abordagens utilizadas para possibilitar a propagação de identidade, e posteriormente a execução de regras de autorização, considera planejamento prévio ao desenvolvimento das aplicações. Estas técnicas tendem a tornar-se trabalhosas e fortemente impactantes quando o objetivo é adaptar sistemas legados e já finalizados, gerando grande gasto de tempo e ocasionando falhas no uso de tais técnicas que possivelmente resultarão em brechas de segurança.

Este trabalho apresentou uma proposta e implementação de uma arquitetura de controle de acesso usando mecanismos de propagação de identidade e de execução de regras de autorização de acesso a dados, considerando um ambiente que já possua soluções para autenticação dos usuários. A técnica gera pouca intrusão em sistemas legados, tais como: (i) aplicação cliente: criação de atributo no ServiceLocator para armazenar informações do usuário e chamada do método para armazenar a identificação do usuário nesta mesma classe, além do armazenamento das informações do usuário no objeto remoto; (ii) objeto remoto: inclusão de atributo para armazenar as informações do usuário e passar as informações do usuário para o HibernateUtil, quando da solicitação de uma sessão que necessite estar autorizada; (iii) HibernateUtil: ajustar as informações do usuário no contexto do SGBD para retornar uma sessão autorizada, quando for o caso. Como as alterações são em partes bem específicas do código e não abrangem todo o código da aplicação, isto diminui a possibi-

lidade de erros, brechas de segurança e gasto de tempo com execução de testes que verifiquem a integridade dos sistemas após as modificações.

Em relação ao desempenho da solução, considerando os aspectos de propagação de identidade, podemos afirmar que o custo é muito baixo, dado que corresponde apenas em passar as informações do usuário necessárias para realizar a autorização. Em geral, é requerido apenas passar a chave do usuário da aplicação do cliente até o SGBD. Já em relação aos aspectos de autorização de acesso (uso do FARBAC), testes experimentais exaustivos foram realizados por Puntar *et al.* (2011). Os resultados obtidos demonstraram que a diferença de tempo entre utilizar o FARBAC em relação ao uso da solução VPD (Virtual Private Database) implementada no Oracle é muito pequena. Além disso, as vantagens de flexibilidade de uso da proposta para gestão das regras de autorização em um local centralizado (todas as funções são armazenadas em um único modelo) demonstram ser viável o uso do FARBAC perante o VPD tradicional.

A implementação empregou uma série de tecnologias disponíveis e de amplo uso no mercado, fazendo com que a técnica possa ser reutilizada em outros cenários. Testes experimentais foram realizados demonstrando a viabilidade e eficácia da proposta em um cenário que simula uma aplicação real de apoio à decisão.

Como trabalhos futuros, propomos a realização de testes de desempenho e avaliação em um cenário real de empresa ou órgão governamental, bem como a evolução da solução para contemplar o quarto cenário apresentado na Seção 2.

REFERÊNCIAS

AZEVEDO, L. G.; PUNTAR, S.; THIAGO, R.; BAIÃO, F.; CAPPELLI, C. A flexible framework for applying data access authorization business rules. In: International Conference on Enterprise Information Systems. 12. *Proceedings...* p. 275-280, May, 2010.

BRG. Defining business rules: what are they really? The Business Rules Group, Jul., 2000. Disponível em: http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf. Acesso em: 15/08/2011.

CALÌ, A.; MARTINENGHI, D. Querying data under access limitations. In: IEEE International Conference on Data Engineering. 24. *Proceedings...* p. 50-59, April, 2008.

DoD, Department of Defense, 1985. Trusted computer security evaluation criteria. Department of Defense, DoD 5200.28-STD. Disponível em: http://csrc.nist.gov/publications/history/dod85.pdf. Acesso em: 15/08/2011.

FERRAIOLO, D. F.; KHUN, D. R. Role-based access control. In: National Computer Security Conference. 15., Baltimore, MD, p. 554-563, *Proceedings...* Oct., 1992.

FERRAIOLO, D. F.; SANDHU, R.; GAVRILA, S.; KUHN, D. R.; CHANDRAMOULI, R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*. v. 4, n. 3, p. 224-274, Aug. 2001. http://dx.doi.org/10.1145/501978.501980

IBM, RACF security guide, IBM, 2010. Disponível em: http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/topic/com.ibm.cics.ts.doc/pdf/dfht5c00.pdf. Acesso em: 15/08/2011.

JELOKA, S.; MULAGUND, G.; LEWIS N. Oracle database security guide, Oracle RDBMS 10gR2. Oracle Corporation, 2008. Disponível em: http://download.oracle.com/docs/cd/B19306_01/network.102/b14266.pdf. Acesso em: 15/08/2011.

LEÃO, F.; PUNTAR, S.; AZEVEDO, L. G.; BAIÃO, F.; CAPPELLI, C. Controle de acesso a dados em sistemas de informação através de mecanismos de propagação de identidade e execução de regras de autorização. In: Simpósio Brasileiro de Sistemas de Informação. 7., p. 117-128, *Proceedings...* Maio, 2011.

MURTHY, R.; SEDLAR, E., Flexible and efficient access control in Oracle. In: ACM SIGMOD 2007, p. 973-980, Beijing, *Proceedings...* Jun., 2007. http://dx.doi.org/10.1145/1247480.1247596

NEEDHAM, R. M.; SCHROEDER, M. D., Using encryption for authentication in large networks of computers. *Communications of the ACM*, v. 21, n. 12, Dec., 1978. http://dx.doi.org/10.1145/359657.359659

PRASANNA, D., R. *Dependency injection*. Manning Publications Co., 2009.

PUNTAR, S.; AZEVEDO, L. G.; BAIÃO, F.; CAPPELLI, C. Implementing flexible and efficient authorization business rules in information systems. In: IADIS International Conference Applied Computing 2011, p. 331-338, Rio de Janeiro, *Proceedings...*, Dec., 2011.

SANDHU, R. S.; COYNE, E. J.; FEINSTEIN, H. L.; YOUMAN, C. E. Role-based access control models. *IEEE Computer*, v. 29, n. 2, p. 38-47, Feb., 1996. http://dx.doi.org/10.1109/2.485845

TPCH. TPC benchmark H standard specification revision 2.8.0. Transaction Processing Perfermance Council. Disponível em: http://www.tpc.org/tpch/spec/tpch2.8.0.pdf. Acesso em: 15/08/2011.

YANG, L. Teaching database security and auditing. In: ACM Technical Symposium on Computer Science Education, v. 40., n. 1, p. 241-245, *Proceedings...* Mar., 2009. http://dx.doi.org/10.1145/1508865.1508954