INSTITUTO DE PESQUISAS TECNOLÓGICAS DO ESTADO DE SÃO PAULO

Jenner Gigante Pereira Borges

Uma ferramenta para a configuração e gerência de papéis(FCGP) em sistemas de bancos de dados

São Paulo

2009

Jenner Gigante Pereira Borges

Uma ferramenta para a configuração e gerência de papéis(FCGP) em sistemas de bancos de dados

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação

Data da aprovação ____/____/_____

_____

Profª. Drª. Edit G. Lino de Campos (Orientadora)
IPT- Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Membros da Banca Examinadora:

Profª. Drª. Edit G. Lino de Campos (Orientadora)
IPT- Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Prof. Dr. João Batista Camargo Junior (Membro)
USP – Universidade de São Paulo

Profª. Drª. Judith Virginia Pavón Mendonza (Membro)
Universidade Anhembi Morumbi

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

B732f Borges, Jenner Gigante Pereira

Uma ferramenta para a configuração e gerência de papéis (FCGP) em sistemas de bancos de dados. / Jenner Gigante Pereira Borges. São Paulo, 2009.

85p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software.

Orientadora: Profa. Dra. Edit G. Lino de Campos

1. Controle de acesso 2. RBAC (Role-Based Access Control) 3. Segurança 4. Banco de dados 5. Tese I. Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Coordenadoria de Ensino Tecnológico II. Título

10-03 CDU 004.632(043)

Jenner Gigante Pereira Borges

Uma ferramenta para a configuração e gerência de papéis(FCGP) em sistemas de bancos de dados

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.

Área de concentração: Engenharia de Software

Orientadora: Profª. Drª. Edit G. Lino de Campos

São Paulo
Novembro/2009

RESUMO

Um dos problemas no gerenciamento de sistemas de banco de dados é a complexidade da administração de segurança. Atualmente a maior parte desta administração é feita por meio do controle de acesso baseado em papéis chamado RBAC (*Role-Based Access Control*), no qual a cada papel é atribuído um conjunto de permissões que representam as ações que os usuários associados a este papel podem executar. Entretanto um dos problemas que podem surgir com este tipo de controle de acesso é o da exclusividade de papéis no qual um mesmo usuário não pode desempenhar dois papéis que poderiam dar-lhe um poder excessivo – como um mesmo usuário cumprindo dois papéis, o de "gerente de compras" e o de "gerente financeiro". Em alguns sistemas de banco de dados não existe um mecanismo que impeça que um mesmo usuário acabe assumindo papéis que deveriam ser mutuamente exclusivos. O objetivo do trabalho é criar uma ferramenta que permita que as regras impostas sobre os relacionamentos entre papéis, conhecidas no modelo RBAC como separações estáticas de deveres, sejam definidas e armazenadas de forma que, por meio delas, o banco de dados possa impedir que ocorram associações indevidas. Hoje, em não existindo este tipo de mecanismo, a verificação muitas vezes é feita por meio de uma série de consultas a tabelas do dicionário de dados e da análise manual dos resultados. A principal contribuição da ferramenta é que ela auxilia a criação inicial destas regras, e uma vez definidas, a única tarefa do administrador é a associação ou não dos papéis aos usuários, ficando a parte de validação a cargo das regras previamente definidas.

Palavras-chave: Segurança, RBAC, separação de deveres, papéis, controle de acesso.

ABSTRACT

Role management and configuration tool for RDBMS

One of the problems in managing database systems is the complexity of security administration. Today security administration is done by the Role Based Access Control standard (RBAC), which recommends that a set of permissions should be associated to a role, indicating the actions that the user assigned to that role can execute. However one of the problems that can occur with this kind of access control is the exclusivity of roles where one user cannot be associated with two roles that could give him excessive power – like the same user being associated with “Purchasing Manager” and “Financial Manager”. In some database systems there is nothing that can avoid mutually exclusive roles of being granted to the same user. The purpose of the this work is to develop a tool that allows to define and manage security policies and restrictions on role relationships as recommended by RBAC’s *static separation of duty*, which prevents the database to allow forbidden relationships. Today, in the absence of this kind of control, this can be done by executing a few queries against the data dictionary and comparing the results, but this process is very time consuming and prone to errors, because it is usually manually done.The main contribution of the tool is that it will help the creation of these relationships and once they are established the main administrative actions will be to associate or disassociate users to roles as job assignments, leaving to the tool the task of enforcing the static separation of duty.

Keywords: Security, RBAC, Separation of Duty, roles, access control.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANSI	*American National Standards Institute*
DAC	*Discretionary Access Control*
DBA	*Database Administrator*
DSD	*Dynamic Separation of Duty*
DDL	*Data Definition Language*
MAC	*Mandatory Access Control*
NIST	*National Institute of Standards and Technology*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
SSD	*Static Separation of Duty*
RBAC	*Role Based Access Control*
FCGP	Ferramenta de Configuração e Gerência de Papéis

Sumário

1 INTRODUÇÃO

1.1 Motivação

Os Sistemas Gerenciadores de Banco de Dados (SGBD) conseguem armazenar e gerir quantidades cada vez maiores de informação e, embora a tornem mais útil e disponível, eles enfrentam problemas de vulnerabilidade no acesso não autorizado. Pensando-se na segurança destes dados faz-se necessária a criação e implementação de mecanismos e políticas de segurança.

Atualmente, para proteger o banco de dados contra ameaças a sua integridade, disponibilidade e confidencialidade, existem medidas como controle de acesso, controle de inferência, controle de fluxo e criptografia. Entre as formas de controle de acesso, a baseada em papéis (*roles*) surgiu como uma tecnologia comprovada para o gerenciamento de segurança em sistemas de grande escala.

O RBAC Ferraiolo (2001) (*Role-Based Access Control*) introduz o conceito de papel (perfil, ou cargo dentro de uma organização), que pode ser associado a um conjunto de ações e responsabilidades. Usuários e operações são associados aos papéis (ao invés de associados diretamente entre si), tornando a manutenção mais consistente e organizada, já que as relações entre permissões de operações e cargos (*roles*) costumam ser mais persistentes que entre permissões e usuários específicos. A cada usuário é associado um ou mais papéis e cada papel está associado às responsabilidades e competências de um determinado cargo dentro da empresa, ou seja, a cada papel estão associados uma ou mais permissões referentes à inclusão, exclusão ou consulta à informação. As operações que um usuário terá permissão de executar estarão relacionadas com os papéis aos quais ele estará associado.

Outra característica do modelo RBAC é sua recomendação quanto ao uso de restrições (*constraints*), as quais podem incidir sobre a autorização de um acesso, permitindo-o ou negando-o com base em critérios mais complexos do que os envolvidos em operações simples. Um exemplo é o da separação de

papéis, em que um mesmo usuário não pode pertencer a dois papéis que poderiam dar-lhe um excesso de poder. Por exemplo, um mesmo usuário participando dos papéis de “gerente de compras” e “gerente financeiro” daria-lhe um excesso de poder que poderia ajudá-lo a praticar ou mascarar fraudes.

Entretanto em virtude de vários sistemas de bancos de dados não implementarem todos os níveis do modelo RBAC, para que problemas como o visto acima sobre a separação de papéis (nível 3 do RBAC) possam ser evitados, uma série de consultas às tabelas do dicionário de dados são necessárias. Por meio destas consultas identificam-se os objetos e permissões de acesso de cada usuário, em seguida é feita uma análise manual desta informação para evitar que um usuário tenha acesso a objetos que lhe permitam executar operações fraudulentas.

Hoje em dia já estão sendo desenvolvidas ferramentas com o propósito de ajudar a solucionar estes problemas em relação a softwares de gerenciamento de sistemas, como o *DirX Identity V7.0* da Siemens DIRX (2004) .Porém quanto a papéis em banco de dados, existem ferramentas que auxiliam na criação e administração de papéis, como o *Oracle Enterprise Manager*, mas que deixam a detecção de sobreposições e de conflitos ainda a cargo dos administradores de banco de dados, sendo muitas vezes feita de forma manual.

1.2 Objetivo

O objetivo do trabalho é desenvolver uma ferramenta para configuração e gerência de papéis (FCGP) em banco de dados segundo o modelo de controle de acesso RBAC (*Role-Based Access Control*). Este modelo trata da segurança de forma aderente à estrutura organizacional de uma empresa. A ferramenta não apenas permite definir os papéis e as permissões dentro de cada papel, mas também permite a associação e dissociação dos papéis aos usuários e a definição de restrições quanto aos relacionamentos entre esses papéis. Estas restrições são necessárias para se evitar que um único usuário, devido aos

papéis aos quais ele esteja associado, seja capaz de, por exemplo, iniciar um processo de pagamento e ter permissões para autorizá-lo.

A ferramenta baseada no controle de acesso RBAC permite que, uma vez que tenham sido definidos os papéis e as permissões associadas a eles, as principais tarefas administrativas sejam apenas a associação e dissociação dos usuários e os respectivos papéis, ficando a cargo da ferramenta a detecção das possíveis violações às restrições de relacionamento entre os papéis.

1.3 Resultados esperados e contribuições

O resultado esperado é o protótipo da ferramenta para a criação e gerenciamento dos papéis em sistemas de bancos de dados que não implementam este nível do modelo RBAC (*constrained* RBAC). A criação e visualização das restrições nos relacionamentos entre papéis possibilitam a identificação automática de falhas e conflitos no momento da associação dos papéis aos usuários.

O trabalho contribui para que outros produtos desenvolvidos sigam e implementem o conjunto de funcionalidades definidas no padrão ANSI do RBAC. Ele ainda contribui para que outras funcionalidades importantes e que ainda não fazem parte do padrão, como a cardinalidade de papéis e restrições temporais de acesso, sejam implementadas.

1.4 Método de trabalho

A pesquisa realizada para o desenvolvimento da ferramenta de gerência de papéis, que tem por objetivo facilitar a administração e controle de acesso aos dados por parte dos administradores, baseia-se em grande parte no estudo do modelo RBAC (*Role Based Access Control*) proposto em *A Proposed Standard for Role Based Access Control* Ferraiolo (2001) e em vários outros artigos sobre RBAC encontrados no site do NIST (*National Institute of Standards and Technology*) tais como, *The NIST Model for Role-Based Acess Control: Towards a Unified Standard* de Sandhu (2001), *Role-Based Access Control*

Models de Sandhu (1996). O livro *Role-Based Access Control* de Ferraiolo (2003) também foi usado como importante fonte de pesquisa.

Neste trabalho são estudados os quatro componentes do modelo RBAC proposto pelo NIST: Núcleo RBAC, RBAC Hierárquico, Separação Estática de Deveres e Separação Dinâmica de Deveres.

O primeiro passo para o desenvolvimento da ferramanta é a definição dos requisitos, principalmente as operações que a ferramenta deverá ser capaz de realizar no gerenciamento de usuários, papéis, permissões e suas respectivas associações. A utilidade destas definições é determinar o escopo de uso da ferramenta e eliminar a ambiguidade no processo de avaliação.

Em seguida especifica-se o projeto de software por meio das descrições, do fluxo de eventos e dos diagramas de casos de uso, do diagrama de classes e do diagrama de sequência em UML. A finalidade deste passo é descrever as várias partes que compõe o software, para auxiliar os desenvolvedores na visualização do protótipo de software em diagramas padronizados.

De acordo com a especificação, inicia-se a fase de desenvolvimento do protótipo em Java, linguagem que oferece portabilidade, ou seja, permite à aplicação ser executada em diferentes arquiteturas, seja de hardware ou software. Com isto, o protótipo poderá funcionar em um sistema operacional que possua a máquina virtual Java ou JVM (*Java Virtual Machine*) instalada, como Windows, Linux, Solaris, AIX. Inicialmente será realizada a construção do código base do protótipo, código fonte que implementa o núcleo das principais funcionalidades do projeto.

Como estudos de caso são examinados os papéis e as restrições sobre os relacionamentos entre papéis de uma instituição financeira. A partir da definição dos papéis e das permissões, são definidos quais papéis devem ou não ser mutuamente exclusivos e então cadastrados pela ferramenta.

1.5 Organização do trabalho

Os próximos capítulos encontram-se organizados conforme descrito a seguir. No capítulo 2, Revisão Bibliográfica, é feita uma contextualização da área com a apresentação dos conceitos relativos a controle de acesso e a mecanismos de controle discricionário e obrigatório.

No capítulo 3, RBAC – *Role-Based Access Control*, é feita a descrição do modelo de controle de acesso RBAC – *Role-Based Access Control*, suas principais características, o mecanismo de controle de acesso baseado em papéis para sistemas de banco de dados, a atribuição e ativação de papéis e o modelo RBAC do NIST (National Institute of Standards and Technology).

O capítulo 4, Projeto e Implementação, é utilizado para definir os requisitos, isto é, quais operações a ferramenta deverá ser capaz de realizar, tais como, manutenção de usuários, de papéis, associações entre papéis, entre permissões e papéis, e entre papéis e usuários. O capítulo mostra o modelo relacional de dados, os algoritmos e as interfaces de usuário utilizados pela ferramenta.

No capítulo 5, Testes da FCGP, são analisados os resultados obtidos no estudo de caso, com aplicações práticas do uso da ferramenta para uma agência bancária. O capítulo mostra também uma comparação no número de associações entre usuários, papéis e permissões, necessárias para se manter uma política de controle de acesso, entre um modelo RBAC e um modelo discricionário.

No capítulo 6, Conclusões, é feita uma análise do trabalho na qual são apresentadas suas contribuições, resultados e sugestões de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 Introdução

Neste capítulo são apresentados conceitos relativos à segurança em bancos de dados e respectivos mecanismos de controle de acesso discricionário, obrigatório e baseado em papéis. Estas definições são importantes para comparar-se os mecanismos e entender-se porque o mecanismo de controle de acesso baseado em papéis está se tornando o mais utilizado. Este último é a base para o desenvolvimento da ferramenta gráfica para gerência de configuração de papéis em bancos de dados.

A segurança em banco de dados é uma área bastante ampla que se refere a questões legais e éticas quanto ao direito de acesso a certas informações, assim como a questões políticas nos níveis governamental, institucional ou corporativo. Estas questões referem-se aos tipos de informação que não devem ser tornadas disponíveis publicamente, como por exemplo, avaliações de crédito e registros médicos pessoais. Referem-se também à sua forma de implementação, isto é, os níveis de sistema nos quais as várias funções de segurança devem ser implementadas, se no nível físico (*hardware*), no nível de sistema operacional ou no nível de SGBD (Sistema Gerenciador de Banco de Dados).

As ameaças aos bancos de dados resultam na perda ou na degradação de alguns ou de todos os seguintes aspectos de segurança Ferraiolo (2007):

Integridade: A integridade do banco de dados refere-se à exigência de que a informação seja protegida contra a modificação imprópria. A modificação de dados inclui a criação, a inclusão, a alteração e a exclusão do dado. A integridade é perdida se modificações não autorizadas são realizadas nos dados tanto por atos intencionais quanto por atos acidentais. Se a perda de integridade do sistema ou dos dados não for corrigida, o uso prolongado do sistema contaminado ou de dados corrompidos pode resultar em imprecisão, fraude ou em decisões erradas.

Disponibilidade: A disponibilidade de um banco de dados refere-se a tornar os objetos disponíveis para um usuário humano ou para um programa que tenha direito legítimo a eles.

Confidencialidade: A confidencialidade de um banco de dados refere-se à proteção dos dados contra a divulgação não autorizada. A divulgação não autorizada, imprevista, ou não intencional pode resultar na perda da confiança pública, em constrangimentos, ou na ação legal contra a organização.

Em um sistema de banco de dados multiusuário, o SGDB deve oferecer mecanismos que habilitam certos usuários ou grupos de usuários a acessar partes selecionadas de um banco de dados sem obter acesso ao restante do banco. Por exemplo, informações sensíveis, como salários de funcionários ou avaliações de desempenho, devem ser mantidas confidenciais em relação à maioria dos usuários do sistema. Tipicamente, um SGDB inclui um subsistema de segurança de banco de dados e de autorização que é responsável por garantir a segurança de partes de um banco de dados contra o acesso não autorizado. Atualmente, os três tipos de mecanismos de controle de acesso em banco de dados mais comuns são Elmasri (2003):

Mecanismos de acesso discricionário: São utilizados para conceder permissões a usuários, inclusive a capacidade de acessar arquivos, registros ou campos específicos de uma maneira específica (como leitura, inclusão, exclusão ou atualização).

Mecanismos de acesso obrigatório: São utilizados para impor a segurança em vários níveis por meio da classificação dos objetos e dos usuários em várias classes de segurança (ou níveis), de acordo com a política de segurança adequada da organização. Por exemplo, uma política de segurança típica é permitir aos usuários em um determinado nível da classificação verem apenas os dados classificados no próprio nível de classificação do usuário (ou abaixo).

Mecanismos de acesso baseado em papéis: São utilizados como uma alternativa viável para os controles de acesso discricionário e obrigatório. Neste

mecanismo as permissões são associadas aos papéis, e aos usuários são atribuídos os papéis adequados.

2.2 Conceitos sobre Controle de Acesso

Uma definição de **acesso,** no contexto deste trabalho, é a possibilidade de realizar alguma operação(visualizar, alterar, etc.) com algum recurso computacional (arquivo, impressora, etc.). Controle de acesso De Capitani (2003) é o meio pelo qual a capacidade de realizar a operação é explicitamente habilitada ou restringida de alguma forma.

2.2.1 Autenticação, autorização e auditoria

O controle de acesso, em segurança da informação, é realizado por meio de processos de autenticação, autorização e auditoria Ferraiolo (2007).

Autenticação: é o processo que verifica se uma identificação fornecida por um usuário é legítima. Em termos gerais, é o processo de provar ao sistema que o usuário é realmente quem diz ser, e não alguém se passando por ele. O modo mais comum de autenticação é por senha em que o usuário autentica-se ao sistema fornecendo-lhe alguma informação que (assume-se) só é de conhecimento do usuário. Formas menos comuns são *smartcards* Asokan (1997) e características biométricas, como padrões de íris, impressão digital e padrões de voz.

Autorização: este processo verifica quais operações e quais recursos computacionais, o usuário poderá utilizar no sistema. Entre as implementações de processos de autorização mais comuns estão o processo de autorização discricionário, obrigatório e o baseado em papéis.

Auditoria: é uma referência à coleta da informação relacionada à utilização, pelos usuários, dos recursos de um sistema. Esta informação pode ser utilizada para gerenciamento, planejamento, cobrança e etc. As informações que são tipicamente relacionadas com este processo são a identidade do usuário, a natureza do serviço entregue, o momento em que o serviço se inicia e o momento do seu término.

2.2.2 Usuário, sujeito, objeto, operações e permissões

A maioria dos modelos de controle de acesso podem ser expressos formalmente usando as noções de usuários, sujeitos, objetos, operações, permissões e as relações entre estas entidades Ferraiolo(2007).

O termo **usuário** refere-se à pessoa que interage com o sistema computacional. À ocorrência de um diálogo entre o usuário e o sistema computacional dá-se o nome de sessão.

Sujeito é um programa em execução, também chamado de processo computacional, que está agindo em benefício de um usuário. Um usuário pode ter múltiplos sujeitos em operação em um certo momento. Por exemplo, um sistema de *e-mail* pode estar operando em segundo plano buscando *e-mails* de um servidor, periodicamente, enquanto o usuário opera um *Web browser*. Cada um destes programas em execução é um sujeito.

Objetos representam qualquer recurso acessível em um sistema, incluindo arquivos, periféricos, bancos de dados ou entidades elementares como campos individuais de um banco de dados. Objetos são tradicionalmente vistos como entidades passivas, que contêm ou recebem informação.

Operações são realizadas pelos sujeitos do sistema sobre objetos. As operações e objetos que o método de controle de acesso gerencia são dependentes do tipo do sistema no qual ele está implementado. As operações mais comuns são leitura, escrita e remoção.

Permissão (privilégio) é o direito de executar uma determinada operação sobre um dado objeto, como por exemplo, permissão de leitura de uma tabela dentro do banco de dados.

Papel (*role*) ou **perfil** é uma coleção de funções que um usuário ou um grupo de usuários pode executar dentro do contexto de uma organização. Usuários e permissões são associados aos papéis ao invés de associados diretamente entre si.

2.2.3 Políticas, Mecanismos e Modelos

No planejamento de um sistema de controle de acesso, três abstrações são consideradas: políticas, mecanismos e modelos de controle de acesso.

Políticas de controle de acesso são requisitos de alto nível que especificam como o acesso é controlado e quem, sob quais circunstâncias, pode acessar quais informações. Estas políticas, por sua vez, são impostas por **mecanismos** que analisam as solicitações de acesso dos usuários utilizando estruturas internas. Por exemplo, uma busca em uma tabela contendo dados sobre usuários, objetos e permissões pode ser realizada para conceder ou revogar o acesso Elmasri (2003).

Um **modelo** de controle de acesso é uma representação formal de uma política de controle de acesso implementada por um sistema por meio de algum mecanismo. Os modelos são úteis para fornecer as limitações teóricas de um sistema. Os usuários visualizam o modelo como uma representação de seus requisitos, enquanto desenvolvedores visualizam o modelo como representação dos requisitos de projeto e implementação. Modelos de controle de acesso descrevem as propriedades de um sistema de controle de acesso. Os mecanismos de controle de acesso são projetados para aderirem às propriedades de um modelo.

2.2.4 Mecanismos e Estruturas para Controle de Acesso

Um dos primeiros trabalhos na definição formal e descrição matemática do controle de acesso foi feito por Lampson (1969) que introduziu as noções formais de sujeito e objeto e uma matriz de acesso que mediava o acesso de sujeitos à objetos.

Matriz de acesso: composta por colunas representando os objetos e linhas representando os sujeitos. Em cada célula representam-se as permissões que o respectivo sujeito possui sobre o objeto. No exemplo da Tabela 2.1, segue-se a convenção do UNIX, em que a letra “R” representa permissão de

leitura, a letra “W”, a permissão de escrita e a letra “X”, a permissão de execução.

Tabela 2. 1- Exemplo de matriz de acesso

	Arquivo de SO	Arquivo de Impressão	Planilha de RH	Planilha Financeira
Sujeito1	R-W-X	R-W-X	R-W	R
Sujeito2	X	X	R-W	-
Sujeito3	R-X	R	R	R

Fonte: Elaborada pelo autor

Matrizes de acesso não são adequadas para implementação direta, pois podem ser muito esparsas.Por exemplo, um sistema com 20.000 usuários e 200 arquivos precisaria de uma matriz com 4 milhões de células, gerando problemas de espaço e maior risco de erros na administração.

Listas de controle de acesso (*ACL- Access Control List*): definem os direitos e permissões que são dados a um sujeito sobre determinado objeto. Esta lista é armazenada junto com cada objeto e relaciona quais permissões cada sujeito possui naquele objeto. O exemplo da Tabela 2.2 mostra uma lista de controle de acesso para a planilha de RH do exemplo anterior.

Uma lista de controle de acesso corresponde a uma coluna da matriz de acessos e refere-se a um objeto.

As listas são utilizadas em sistemas operacionais como Unix e Windows, bem como em roteadores e *softwares* de *firewall*.

Tabela 2. 2 - Lista de controle de acesso para um objeto

	Planilha de RH
Sujeito1	R-W
Sujeito2	R-W
Sujeito3	R

Fonte: Elaborada pelo autor

Bits de proteção (*Protection bits*): O mecanismo de bits de proteção Elmasri (2003) esta incluído nos sistemas UNIX e similares. Este mecanismo utiliza apenas alguns bits anexados a cada arquivo, que especificam as permissões de leitura, escrita e execução para as diferentes classes de usuários. Estes bits contem informação sobre:

- O proprietário do arquivo.

- Grupo de usuários pertencendo ao mesmo grupo do proprietário do arquivo.

- Outros: quaisquer outros usuários ou grupos diferentes dos acima citados.

O sistema de controle de acesso regula o acesso ao arquivo por meio da associação das operações de leitura(R), escrita(W) e execução(X) a cada uma destas categorias, como mostra da Tabela 2.3.

Tabela 2. 3 - Bits de permissão

Proprietário			Grupo			Outros		
R	W	X	R	W	X	R	W	X

Fonte: Elaborado pelo autor

2.3 Políticas de Controle de Acesso: DAC e MAC

O controle de acesso discricionário(*discretionary access control* ou DAC) é uma política de controle de acesso determinada pelo proprietário (*owner*) do objeto (um arquivo, por exemplo). O proprietário do objeto decide quem tem permissão de acesso e quais permissões ele possui. Um usuário transforma-se em proprietário do objeto ao criá-lo. O DAC tem dois conceitos importantes:

- Todo objeto em um sistema deve ter um proprietário. A política de acesso é determinada pelo proprietário do recurso.

- Permissões são dadas aos usuários pelos proprietários dos objetos.

No controle de acesso obrigatório (*mandatory access control* ou MAC) a política de acesso é determinada pelo sistema e não pelo proprietário do objeto. Este controle é utilizado em sistemas de múltiplos níveis de segurança cujos dados são altamente sensíveis, como algumas informações governamentais e militares. Para cada objeto e usuário do sistema é associado um nível de segurança. O nível de segurança associado ao objeto reflete o nível de importância da informação contida no objeto, classificando o objeto quanto ao dano potencial que um acesso não autorizado traria. O nível de segurança associado ao usuário, também chamado de *clearance,* reflete o nível de confiabilidade do usuário em não expor a informação a um usuário que não esteja autorizado para tal Sandhu (1994). O acesso de um sujeito (*subject*) a um objeto é verificado em função do relacionamento entre os níveis de segurança deles e levando-se em conta o tipo de acesso solicitado.

2.3.1 Política de controle de acesso discricionário (DAC).

Sempre que uma pessoa ou grupo de pessoas precisa acessar um sistema de banco de dados, a pessoa ou o grupo deve primeiro requisitar uma conta de usuário (*user account)*, que, muitas vezes, é implementada internamente como um número. Quando o usuário tenta se conectar (*login*) ao SGBD, o sistema verifica se a conta e a senha são válidos; caso sejam, o usuário tem a permissão de acessar o banco de dados Ferraiolo (2003) de acordo com suas respectivas permissões (privilégios).

O SGBD por meio da linguagem SQL3 suporta o DAC com a utilização dos comandos *GRANT* and *REVOKE*. Existem dois níveis para a atribuição de permissões para o uso do sistema de banco de dados:

Nível de conta: Nesse nível, o administrador de banco de dados (DBA) estabelece as permissões específicas que cada conta deve ter, independentemente das tabelas no banco de dados.

Nível de tabela (ou relação): Nesse nível, o DBA pode controlar a permissão para acessar cada tabela ou visão (*view*) individual no banco de dados.

As permissões no nível de conta aplicam-se às capacidades providas para a própria conta e podem incluir, por exemplo, as permissões CREATE TABLE, CREATE VIEW, ALTER, DROP, MODIFY e o SELECT. Essas permissões de conta se aplicam à conta de maneira genérica. Se uma certa conta não possuir a permissão CREATE TABLE, nenhuma tabela pode ser criada a partir daquela conta.

O segundo nível de permissões se aplica ao nível de tabela, seja ela uma tabela base ou uma tabela virtual (visão ou *view*). As permissões no nível de tabela especificam, para cada usuário, as tabelas individuais às quais cada comando pode ser aplicado. Algumas permissões também se referem a colunas individuais (atributos) de tabelas.

As permissões que podem ser concedidas para uma tabela (relação) R individual:

- *SELECT sobre R:* Possibilita à conta de usuário a permissão de acesso para busca das tuplas (linhas) de R.
- *MODIFY sobre R:* Proporciona à conta de usuário a capacidade de modificar as tuplas de R. Este privilégio é dividido nas permissões UPDATE, DELETE e INSERT. Além disso, as permissões INSERT e UPDATE podem especificar que apenas certos atributos de R podem ser atualizados.
- *REFERENCES sobre R:* Proporciona à conta de usuário a capacidade de fazer referência à relação R quando estiver especificando relações de integridade.

Além da concessão das permissões acima, feita pelo comando *GRANT*, pode-se revogá-los usando o comando *REVOKE*.

Uma outra característica do mecanismo de controle de acesso discricionário para sistemas de banco de dados é a propagação de permissões com o uso da opção *GRANT OPTION*. Uma permissão que é concedida com a opção *GRANT OPTION* pode ser passada para outros usuários sem conhecimento do proprietário do objeto. Assim, sempre que o proprietário U1 de uma relação R concede uma permissão sobre R para uma outra conta U2, a permissão pode ser dada para U2 com ou sem o *GRANT OPTION*. Se o *GRANT OPTION* é dado, então U2 também pode conceder aquela permissão sobre R para outras contas. Dessa maneira, as permissões sobre R podem propagar-se para outras contas sem o conhecimento do proprietário de R. Se a conta U1 revogar a permissão concedida a U2, todas as permissões que U2 tenha propagado devem ser automaticamente revogadas pelo sistema.

As principais desvantagens deste mecanismo são:

- A informação pode ser copiada de um objeto (tabela) para outro, de modo que o acesso a uma cópia é possível mesmo que o proprietário do objeto original não tenha concedido esta permissão. Por exemplo, quando um usuário U1, proprietário de uma tabela T1, concede direitos de leitura em T1 ao usuário U2, nada impede o usuário U2 de copiar o conteúdo da tabela do usuário U1 para dentro de uma tabela que o usuário U2 controle. O usuário U2 pode agora conceder qualquer permissão à sua cópia, sem o conhecimento do usuário U1.

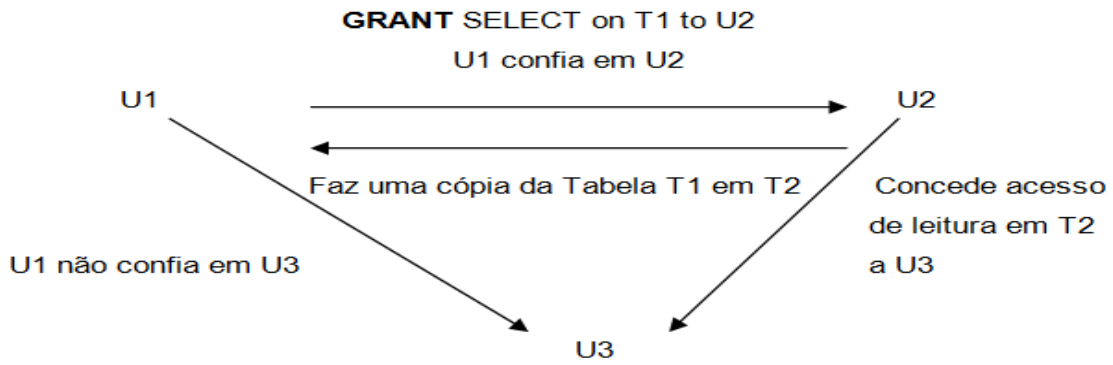

Figura 2. 1 - Exemplo de falha na relação de confiança (DAC)

Fonte: Elaborado pelo autor

- O proprietário do objeto decide quais permissões pode conceder sobre o objeto, ao invés destas permissões serem atribuídas por um sistema central de controle que reflita os requisitos de segurança da organização.

- O mecanismo é vulnerável a ataques de programas chamados de "Cavalos de Tróia". Um Cavalo de Tróia (*Trojan Horse*) é um programa que além de executar funções para as quais foi aparentemente projetado, também executa outras funções normalmente maliciosas e sem o conhecimento do usuário. Devido ao fato de que os programas assumem a identidade do usuário que os executam, o usuário U3 pode escrever um programa para o usuário U1, que a princípio executa alguma operação útil, mas que ao mesmo tempo lê o conteúdo das tabelas do usuário U1, sem seu conhecimento, e escreve o seu conteúdo em tabelas pertencentes ao usuário U3. O programa "Cavalo de Tróia" do usuário U3 pode até mesmo remover linhas das tabelas de U1 que as trilhas de auditoria indicariam como tendo sido removidas por U1.

2.3.2 Política de controle de acesso obrigatório (MAC).

No controle de acesso obrigatório (*mandatory access control* ou MAC) a política de acesso é determinada pelo sistema e não pelo proprietário do objeto. Em muitas aplicações governamentais, militares e de inteligência, uma política de segurança adicional, que classifique objetos e sujeitos baseando-se em classes ou níveis de segurança(segurança multi-nível), é necessária. Assim, rótulos de segurança são associados a usuários, sujeitos e objetos dentro do sistema. Os usuários não são considerados donos dos objetos e não podem definir suas permissões, isto é realizado pelos administradores do sistema.

A política de acesso mandatória definida por Bell-LaPadula Bell(1973) para sistemas militares norte-americanos é expressa em termos de rótulos ou níveis de segurança. Estes níveis de segurança possuem um componente hierárquico e um não-hierárquico. Por exemplo, uma organização militar pode ter os componentes hierárquicos "Não Confidencial"(NC), "Confidencial"(C), "Secreto"(S) e "Altamente Secreto"(AS), enquanto os componentes não-

hierárquicos podem ser “Nuclear”, “Militar” e “NATO”(*North Atlantic Treaty Organization*), identificando as áreas ou categorias das informações classificadas . Os níveis de segurança são parcialmente ordenados sob uma relação de domínio, descritas por “ ≥ ”. Por exemplo, considerando-se a seguinte ordem, AS ≥ S ≥ C ≥ NC e S(Militar,Nuclear) ≥ S(Nuclear), dados dois níveis de segurança N1 e N2 , diz-se que N1 **domina** N2, N1 ≥ N2, se e apenas se o componente hierárquico(AS,S,C e NC) de N1 é maior ou igual ao de N2, e o componente não hierárquico(Nuclear,Militar,NATO) de N1 inclui o componente de N2.

O nível de segurança do usuário reflete o nível de confiança dado a ele e deve sempre dominar os níveis de segurança que são associados aos sujeitos deste usuário. Por exemplo, um usuário chamado João, com nível de segurança S(Nuclear) pode iniciar sessões nos níveis S(Nuclear), S, C, ou NC. Os níveis de segurança associados aos objetos refletem a sensibilidade do conteúdo dos objetos. Com relação ao nível de segurança de um sujeito e de um objeto, o modelo Bell-LaPadula Bell (1973) define decisões de controle de acesso baseadas em duas propriedades:

- ***Simple-Security Property (“no read up”)***: Um sujeito tem permissão de leitura em um objeto se o nível de segurança do sujeito **domina** o nível de segurança do objeto.

- ***Star-Property (“no write down”)***: Um sujeito tem permissão de escrita (inserção, remoção e exclusão) em um objeto se o nível de segurança do objeto domina o nível de segurança do sujeito.

A satisfação destas propriedades impede usuários de lerem informações que **dominam** (estão acima ou são iguais) seus níveis de segurança. A propriedade *star-property* impede sujeitos de escreverem o conteúdo de objetos que estão em um nível X para dentro de objetos que poderiam ser subsequentemente lidos por um sujeito com um nível de segurança que é **dominado** (está abaixo ou é igual) por X. A consequência direta da existência

desta propriedade é a proteção contra o roubo de informação por programas "Cavalos de Tróia".

Com relação ao Modelo Bell-LaPadula Bell (1973) é importante distinguir um usuário de seus sujeitos. Considere um usuário João com nível de segurança AC (Altamente Secreto), este usuário pode ler e escrever todos os objetos que o seu nível de segurança **domina**. Os sujeitos do usuário João, por sua vez, não possuem esta mesma liberdade. Embora se confie em João para não transmitir informações altamente secretas, não se deve atribuir a mesma confiança para os sujeitos de João, pois estes são em sua maioria programas de computadores, que podem estar comprometidos, como um programa "Cavalo de Tróia". Portanto, para João conseguir escrever em um objeto que esteja, por exemplo, no nível Secreto (S), João deve ajustar sua sessão para um nível que é **dominado** pelo nível de segurança do objeto.

Por exemplo, João que possui agora nível de segurança confidencial (C), deseja "roubar" informações Secretas e Nucleares usando um programa "Cavalo de Tróia" que será executado por Maria que tem nível de segurança S (Nuclear) e estará executando o programa com uma sessão no seu nível máximo permitido S (Nuclear). Embora o programa de João consiga ler as informações secretas e nucleares de acordo com a propriedade *simple-security,* o programa falhará na sua tentativa de escrever as informações em um objeto ao qual João terá acesso.

2.3.2.1 Relações Multiníveis e Poli-Instanciação.

Para incorporar as noções de segurança em múltiplos níveis do modelo MAC no modelo de banco de dados relacional, cada objeto deve estar associado a um nível de segurança. Os objetos podem estar na granularidade de tabelas, linhas ou colunas. Por exemplo, para cada atributo A de uma tupla é associado um atributo de nível de segurança X. Assim, uma relação de múltiplos níveis de segurança R com n atributos seria representada como:

R(A1,X1,A2,X2,...,An,Xn)

na qual cada X*i* representa o atributo de nível de segurança associado ao atributo A*i*. Uma relação com múltiplos níveis de segurança, relação multinível, parecerá conter dados diferentes para sujeitos com *clearance* diferentes.

Um exemplo de relação multinível pode ser vista na Figura 2.2(a) em que os valores dos atributos de nível de segurança estão ao lado de cada valor de atributo. Considere-se a consulta SELECT * FROM EMPREGADO. Um usuário com nível de segurança S veria a mesma relação mostrada na Figura 2.2(a), uma vez que todos os níveis de segurança dos atributos são menores ou iguais a S. Entretanto, um usuário com nível de segurança C não teria permissão para ver os valores do salário de Carlos e desempenho de função de Antonio, uma vez que eles tem um nível de segurança mais alto. As tuplas seriam filtradas para aparecerem conforme mostradas na Figura 2.2(b), com Salário e Desempenho de função aparecendo como valores *null*.

(a) Relação "Empregado"

Nome	Salário	Desempenho de função
Antonio NC	40 C	Regular S
Carlos C	80 S	Bom C

(b) Relação "Empregado"

Nome	Salário	Desempenho de função
Antonio NC	40 C	*null* S
Carlos C	*null* S	Bom C

(c) Relação "Empregado"

Nome	Salário	Desempenho de função
Antonio NC	40 C	Regular S
Antonio NC	40 C	Excelente C
Carlos C	*null* S	Bom C

Figura 2. 2 - Exemplo de um relação multinível. (a) As tuplas originais de Empregado. (b) Aparência de Empregado após a filtragem para usuários com nível de segurança C. (c) Exemplo de Poli-Instanciação
Fonte: Elmasri (2003)

A principal desvantagem deste mecanismo é que as operações básicas de atualização do modelo relacional (*insert*, *delete*, *update*) precisam ser modificadas. Por exemplo, se um sujeito com *clearance* de segurança C tentar atualizar o valor do atributo “Desempenho de função” de Antonio na Figura 2.2(a) para “Excelente”, uma vez que a visão fornecida aos sujeitos com *clearance* de segurança C permite tal atualização, o sistema não pode rejeitá-la. Caso contrário, o sujeito poderia inferir que algum valor diferente de *null* existe para este atributo. Entretanto, ao sujeito não pode ser permitido sobrescrever o valor existente de “Desempenho de função” sob o risco de perda da integridade, então a solução é criar uma outra tupla para Antonio em um nível de segurança mais baixo C, conforme mostrado na Figura 2.2 (c). A presença de objetos (tuplas no exemplo acima) que aparentam ter valores diferentes para usuários com diferentes níveis de segurança é chamada de Poli-Instanciação.

2.4 Comparação entre controle de acesso discricionário e controle de acesso obrigatório

As políticas de controle de acesso discricionário (DAC) são caracterizadas por uma alta flexibilidade, o que as torna adequadas para uma grande variedade de aplicações. A principal desvantagem do modelo DAC são as suas vulnerabilidades a ataques como os “Cavalos de Tróia” embutidos em programas de aplicação. A razão é que os modelos de autorização discricionários não impõem nenhum controle sobre como a informação é propagada e usada, tendo sido ela acessada por usuários autorizados a fazê-lo. Um mecanismo DAC permite que usuários concedam ou revoguem acesso a quaisquer objetos sob seu controle sem a intercessão de um administrador global do sistema.

Políticas obrigatórias (MAC) asseguram um alto grau de proteção, pois elas previnem qualquer fluxo ilegal de informação. Entretanto, têm a desvantagem de serem muito rígidas na medida em que exigem uma classificação meticulosa de sujeitos e objetos em níveis de segurança.

2.5 Conclusões

Os modelos de segurança são essenciais para a definição de políticas de segurança em um ambiente computacional. Até os dias de hoje, diversos modelos de segurança foram propostos, baseando-se em premissas variadas e visando atender diferentes aspectos de segurança.

Este capítulo mostrou as diferenças básicas entre modelos, políticas e mecanismos de segurança além de políticas de segurança consideradas clássicas na área de segurança computacional e que ainda hoje são largamente utilizadas nos mais variados ambientes.

O capítulo mostrou ainda as definições de usuário, sujeito, objeto, operações e permissões que são entidades utilizadas para se expressar a maioria dos modelos de controle de acesso.

3 RBAC: *Role-Based Access Control*

3.1 Introdução

As raízes do RBAC incluem o uso dos grupos em sistemas operacionais e o controle de permissões nos sistemas gerenciadores de banco de dados nos primeiros sistemas computacionais multiusuários interativos no início da década de 70 Samarati (2001).

Nas décadas de 80 e 90, pesquisadores começaram a reconhecer as virtudes dos papéis como uma abstração para o gerenciamento de permissões dentro de aplicações e sistemas de bancos de dados. Um papel era visto como um cargo ou uma posição dentro da organização. Por exemplo, aplicações bancárias *on-line* naquela época já incluíam os papéis *teller* (caixa de banco) e *teller supervisor* (supervisor de caixa de banco) que podiam executar diferentes conjuntos de transações, enquanto usuários, simultaneamente, por meio de *ATM*s, eram ainda capazes de executar outro conjunto de transações nas mesmas bases de dados. Estes sistemas baseados em papéis eram relativamente simples e específicos de suas aplicações, ou seja, não havia um modelo de propósito geral definindo como o controle de acesso podia ser baseado em papéis. Eles eram desenvolvidos por uma variedade de organizações que não seguiam nenhum padrão ou definição formal.

Em 1992, o NIST (*National Institute of Standards and Technology*) iniciou um estudo Ferraiolo (1993) em organizações comerciais e governamentais e constatou que suas necessidades de controle de acesso não estavam sendo satisfeitas por produtos no mercado, muitos dos quais implementavam apenas controles discricionários. Em muitas organizações os usuários finais não são os proprietários da informação para as quais eles possuem direitos de acesso, como suposto no DAC. Para estas organizações, a empresa ou corporação é o verdadeiro proprietário dos objetos do sistema, sugerindo assim a utilização de um controle como o MAC. Entretanto, mesmo para estas empresas, a implementação do MAC, que faz uso de um esquema de classificação em níveis de segurança, se mostrava muito rígida e trabalhosa.

Uma solução para atender estas necessidades foi proposta em 1992 por Ferraiolo (1992), integrando características específicas de aplicações existentes em um modelo de RBAC geral. O artigo proposto descreveu, de uma maneira simples, os conjuntos, relações e mapeamentos usados nas definições dos papéis (*roles),* como mostrado na Figura 3.1:

Descrição formal original do RBAC
Para cada sujeito, o papel ativo é aquele que o sujeito está usando naquele momento: PA(s : sujeito) = {papel ativo do sujeito s}
Cada sujeito pode ser autorizado a agir como um ou mais papéis: PAu(s : sujeito) = {papéis autorizados para sujeito s}
Cada papel pode ser autorizado a realizar uma ou mais transações: TA(p : papel) = {transações autorizadas para papel p}
Sujeitos podem executar transações. O predicado exec(s, t) é verdadeiro se, e somente se, o sujeito s pode executar a transação t naquele momento; caso contrário, é falso: exec (s : sujeito, t : transação) = {verdadeiro ⇔ sujeito s pode executar transação t} Um sujeito pode executar uma transação somente se o sujeito selecionou ou foi designado a um papel: s : sujeito, t : transação . exec(s,t)-> PA(s)≠ Ø O papel ativo de um sujeito deve estar autorizado para aquele sujeito: PA(s) C PAu(s) Um sujeito pode executar uma transação somente se a transação está autorizada para o papel ativo do sujeito s : sujeito, t : transação . exec(s,t) => t € TA(PA(s)) É importante notar que está última regra condicional permite que outras restrições possam ser colocadas na execução de transações.

Figura 3. 1 - Descrição formal do RBAC

Fonte: Ferraiolo (2003)

Em 1996, Sandhu (2001) introduziu um *framework* de modelos RBAC, RBAC96, quebrando o RBAC em quatro modelos conceituais. Este *framework*

especificou um modelo base, RBAC0, que contém as características mínimas de um sistema implementando RBAC, dois modelos avançados, RBAC1 e RBAC2, que incluem o RBAC0 e adicionam suporte para hierarquias e restrições de papéis respectivamente. Um quarto componente, RBAC3, que inclui todos os aspectos dos modelos de níveis mais baixos.

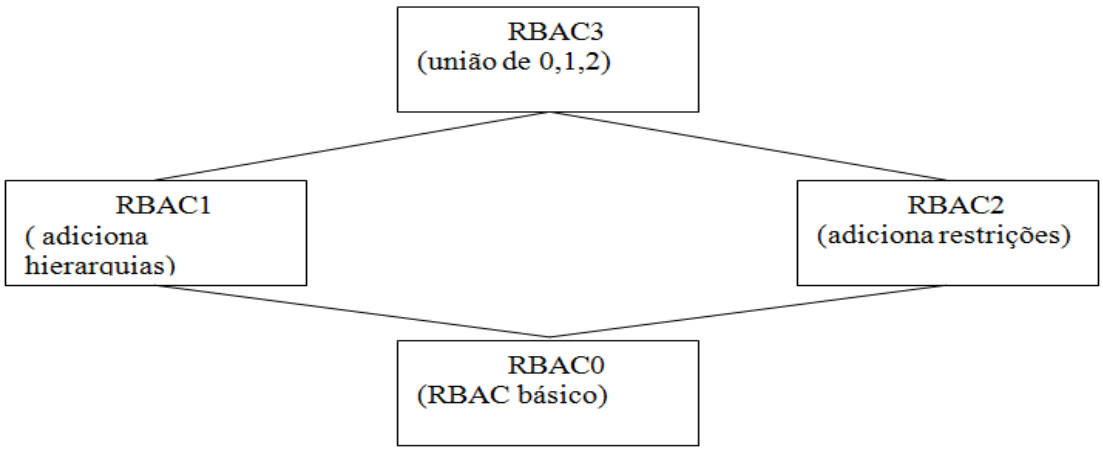

Figura 3.2 - *Framework* proposto por Sandhu
Fonte: Sandhu (2001)

Em 2000, o NIST iniciou um esforço para estabelecer um padrão de consenso internacional para o RBAC, publicando uma proposta Sandhu (2001) no *workshop* ACM RBAC.

O padrão proposto segue a estrutura do RBAC96 e incorpora características advindas de discussões posteriores e comentários recebidos das comunidades de pesquisa e empresas comerciais. Em 2004, o padrão foi aprovado como INCITS 359-2004 pelo *International Committee for Information Technology* que é autorizado pelo *American National Standards Institute* para estabelecer padrões na área de tecnologia da informação.

3.2 O RBAC

No modelo RBAC, os usuários são feitos membros de **papéis**, os quais têm direitos de acesso. Ele usa o conceito de **operações**, representando as ações que um papel pode executar sobre os objetos, e o conceito de **usuários**,

que representam os membros associados aos papéis. A Figura 3.3 mostra esta situação.

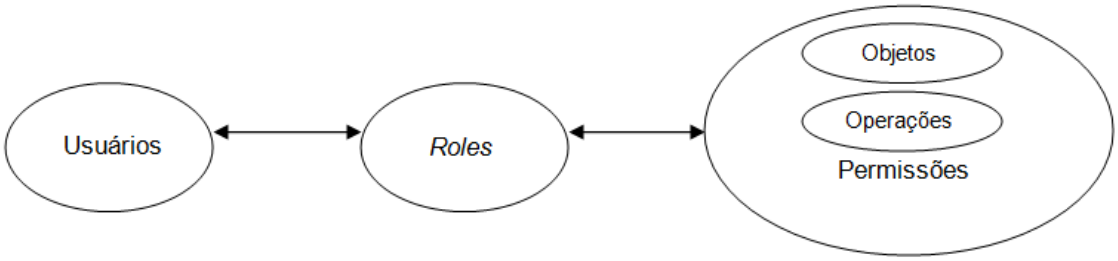

Figura 3. 3 - Relacionamento entre usuários, papéis e permissões
Fonte: Ferraiolo (2003)

Um usuário pode assumir um ou mais papéis e um papel pode ser assumido por um ou mais usuários, como indicado pelas setas duplas na Figura 3.3. As operações podem ser associadas ou excluídas dos papéis objetivando atender às eventuais alterações das atividades desempenhadas e representadas por um papel dentro da organização. Os papéis podem ser atualizados sem a necessidade da modificação individual de cada usuário ou grupos de usuários. As alterações passam automaticamente a serem refletidas em todos usuários que possuem associação com o papel.

Um dos principais aspectos motivadores para o uso do RBAC é o conceito da separação de tarefas entre papéis (*Separation of duties*). Este conceito consiste em dividir tarefas que podem apresentar conflitos de interesses em várias subtarefas executadas por diferentes usuários Brewer(1989).Por exemplo, suponha-se uma política de autorização construída no cenário de uma agência bancária. Neste caso, podem existir papéis como *Atendente, Supervisor* e *Auditor* e operações como *Agendar Operação de Crédito, Autorizar Operação de Crédito* e *Revisar Operações*. Imaginem-se as associações entre operações e papéis mostradas na Tabela 3.1. Duas questões podem ser observadas: 1) Um funcionário da agência bancária que assume os papéis *Atendente* e *Supervisor* jamais poderá assumir o papel *Auditor*, uma vez que este último tem por atribuição revisar operações efetuadas; 2) Durante uma sessão não pode ser permitido que um usuário assuma simultaneamente os

papéis *Atendente* e *Supervisor*, pois este último tem por objetivo autorizar operações de crédito agendadas pelo *Atendente*. Neste caso, o RBAC permite que se estabeleça um relacionamento de separação de tarefas entre estes papéis para evitar que os casos 1 e 2 possam ocorrer.

Tabela 3.1 - Exemplo de política RBAC

Papéis	Funcionários	Operações
Atendente	A	Agendar Operação de Crédito
Supervisor	B	Autorizar Operação de Crédito
Auditor	C	Revisar as Operações

Fonte: Elaborado pelo autor

Além desta facilidade de gerenciamento, o *framework* proposto pelo RBAC é construído sob o princípio do Privilégio Mínimo (*Least Privilege)*, ou seja, um usuário não precisa ser autorizado a executar operações além das tarefas necessárias para desempenhar sua função dentro da organização. Por exemplo, referindo-se novamente à Tabela 3.1, em uma sessão onde um usuário assume o papel *Atendente* ele somente poderá realizar a operação *Agendar Operação de Crédito*.

Um importante conceito também empregado pelo RBAC é a hierarquia entre papéis. Em um cenário típico de uma organização, indivíduos que desempenham diferentes funções podem executar tarefas em comum. Neste contexto, a hierarquia entre papéis simplifica bastante a descrição de direitos de usuários, pois evitaria a necessidade de especificar de forma repetida operações em comum. Ainda, este relacionamento hierárquico pode ser balizado pela estrutura hierárquica das próprias organizações, permitindo que a política de segurança seja descrita de maneira extremamente natural. Em uma hierarquia entre papéis, um papel pode conter outros papéis. No RBAC, um papel hierarquicamente superior pode conter um ou mais papéis hierarquicamente inferiores. Isto significa que um papel superior tem todos as permissões e pode

executar todas as operações que os papéis de níveis inferiores contêm. A Figura 3.4 ilustra este conceito.

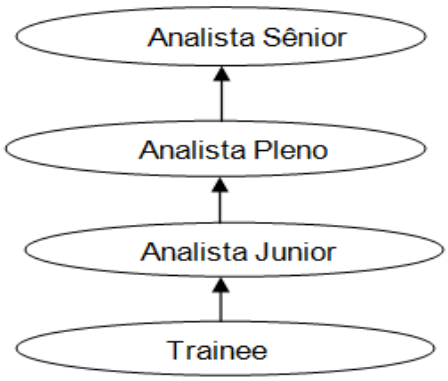

Figura 3. 4 - Exemplo de hierarquia de papéis
Fonte: Elaborado pelo autor

O papel analista pleno, por exemplo, pode executar todas as operações que o papel analista junior pode, ocorrendo o mesmo caso entre os papéis analista sênior e analista pleno. Neste exemplo, o analista pleno herda as permissões do analista junior e o analista sênior os do analista pleno. Obviamente, cada papel, além das permissões herdadas, possui suas próprias permissões associadas.

No RBAC, as políticas são descritas em termos de usuários (*users*), sujeitos (*subjects)*, papéis (*roles*) e operações (*operations*). Para que um dado usuário ganhe acesso a um objeto controlado pelo RBAC, inicialmente ele deve ser incluído como membro de um papel e, em seguida, o papel deve ser ativado para aquele usuário. *Subjects* representam os processos ativados pelo usuário. Conseqüentemente, diversos *subjects* podem ser associados a um usuário.

O modelo RBAC pode ser descrito em termos de primitivas Ferraiolo (1995), que agem como interface ao modelo, e por regras, que definem precisamente as propriedades do modelo. A Tabela 3.2 mostra um resumo das primitivas que compõem a definição do modelo.

Tabela 3. 2 - Primitivas de acesso ao modelo RBAC

Primitiva	Descrição
subject-user(s:subject)	Retorna o usuário associado ao subject s.
authorized-roles(s:subject)	Retorna os papéis associados ao subject s.
role-members(r:roles)	Retorna os usuários autorizados para o papel r.
user-authorized-roles(u:user)	Retorna os papéis associados ao usuário u.
role-operations(r:roles)	Retorna as operações associadas ao papel r.
operation-objects(op:operation)	Retorna os objetos pelos quais a operação op pode ser aplicada.
mutually-exclusive-authorization(r:roles)	Retorna a lista de papéis mutuamente exclusivos com r.
membership-limit(r:roles)	Retorna o número limite de usuários que podem receber o papel r.
number-of-members(r:roles)	Retorna o número de membros existentes no papel r.
exec(s:subject,op:operation)	Retorna true: O processo ativo s pode executar a operação op; false: não pode.
active-roles(s:subject)	Retorna a lista de papéis ativos para o subject s.
mutually-exclusive-activation(r:roles)	Retorna a lista de papéis mutuamente exclusivos dinamicamente com r.
function-operations(f:function)	Retorna o conjunto de todas as operações necessárias para executar a função de negocio f.
access(s:subject,o:object)	Retorna true: O processo ativo s pode acessar o objeto o; false: não pode

Fonte: Ferraiolo (1995)

Para completar a definição concisa do modelo RBAC, a Tabela 3.3 relaciona as principais regras associadas ao modelo.

Tabela 3.3 - Regras de definição do modelo RBAC

Regra	Descrição
1*) Role Hierarchy* $\forall$ s:subject, $r_{i,j}$: roles $r_j \in$ authorized-roles(s) Λ $r_j > r_i$ $\Rightarrow$ $r_i \in$ authorized-roles(s)	Caso um papel rj contenha um papel ri e rj esteja no conjunto de papéis autorizados ao subject s então ri também está neste conjunto.
2*) Static Separation of Duty* $\forall$ u:user, $r_{i,j}$: roles : $i \neq j$: $u \in$ role-member(r_i) Λ $u \in$ role-member(r_j) $\Rightarrow$ $r_i \notin$ mutually-exclusive-authorization(r_j)	Alguns papéis não podem ser assumidos simultaneamente por um mesmo usuário, assim, são mutuamente exclusivos.
3) *Cardinality* $\forall$ r:roles membership-limit(r) $\geq$ number-of-members(r)	Esta regra indica que o número de membros de um determinado papel não pode ultrapassar o número máximo de membros permitidos para ele.
4) *Role Authorization* $\forall$ s:subject active-roles(s) $\subseteq$ authorized-roles(s)	Esta regra indica que um determinado subject s não pode ter um papel ativo que não seja autorizado para ele.
5) *Role Execution* $\forall$ s:subject, op: operation: exec(s,op) $\Rightarrow$ active-roles(s) $\neq \varnothing$	Esta regra indica que um determinado subject s pode executar uma dada operação op se ele está atuando dentro de um papel ativo.
6) *Dynamic Separation of Duty* $\forall$ s:subject, $r_{i,j}$: roles : $i \neq j$: $r_i \in$ active-roles(s) Λ $r_j \in$ active-roles(s) $\Rightarrow$ $r_i \notin$ mutually-exclusive-activation(r_j)	Esta regra indica que um determinado subject s pode se tornar ativo em outro papel desde que este novo papel não seja mutuamente exclusivo com qualquer um dos papéis no qual o subject s está correntemente ativo.
7) *Operation Authorization* $\forall$ s:subject, op: operation, r:roles: exec(s,op) $\Rightarrow$ r $\in$ active-roles(s) Λ op $\in$ role-operations(r)	Esta regra indica que um determinado subject s pode executar uma determinada operação somente se ela está autorizada para o papel no qual o subject s está correntemente ativo.

Fonte: Ferraiolo (1995)

As regras foram construídas baseadas nas primitivas relacionadas na Tabela 3.2. A regra 1 trata da descrição da hierarquia de papéis, indicando a característica de herança entre eles. As regras 2 e 3 estão relacionadas à associação de usuários a papéis e estão baseadas no fato de que não pode ser dada mais permissão do que o necessário a um usuário, de que não pode haver conflito entre os papéis assumidos por um usuário e de que, caso exista uma limitação numérica com relação à atribuição de papéis, ela não pode ser excedida. As regras 4,5,6 e 7 tratam do aspecto de ativação de papéis.

Com intuito de mostrar como políticas são mapeadas por meio do RBAC, a seguir, baseado na hierarquia de papéis mostrada na Figura 3.4 e nas definições indicadas pelas Tabelas 3.2 e 3.3 , será mostrado na Tabela 3.4 um exemplo simplificado da descrição de uma política RBAC.

Tabela 3. 4 - Exemplo de política RBAC

Users: Roberto, Ana**,** Paulo *Subjects:* Projetar Sistema, Analisar Sistema, Programar Sistema *Roles:* Senior, Pleno, Junior e Trainee. *Operations:* Liderar projeto, Orientar analista, Aprovar projeto, Programar módulo do sistema , Analisar sistema *Objects:* Módulo do sistema, Projeto *subject user* (Analisar Sistema): Roberto, Ana *subject user* (Projetar Sistema) : Roberto, Ana**,** Paulo *subject user* (Programar Sistema): Paulo *authorized-roles* (Analisar Sistema): Pleno *role-members* (Senior): Ana, Roberto *role-Operations* (Pleno): Orientar analista junior,Analisar sistema *mutually-exclusive-authorization* (Trainee): Senior *membership-limit* (Senior) = 2 *membership-limit* (Pleno) = 2

Fonte: Elaborado pelo autor

3.3 Modelo RBAC do NIST

O RBAC possui uma série de funcionalidades e propriedades não cobertas pelos modelos tradicionais de controle de acesso. Se isto por um lado aumenta a riqueza e a capacidade da descrição de políticas de segurança, por outro lado aumenta a complexidade da formalização do próprio modelo.

Esta característica cria a necessidade de refinamentos e de divisão do modelo em submodelos, para que seja possível explorar as várias dimensões do RBAC. Sandhu (2001) discutem as várias formas de RBAC, objetivando estabelecer um modelo de referência com a padronização de terminologias e funcionalidades para que desenvolvedores de aplicações com mecanismos RBAC embutidos possam especificar seus produtos e os consumidores de TI possam avaliar e comparar estes produtos. Sandhu (1996) define uma família de

quatro modelos conceituais, os quais foram chamados de RBAC0, RBAC1, RBAC2 e RBAC3 (ver Figura 3.2). Sandhu (2001) propõe uma padronização sem introduzir novas características ao modelo, mas apenas levando em consideração todo avanço conseguido com a publicação dos diversos trabalhos da área, dando enfoque à definição dos componentes fundamentais do RBAC. Seu trabalho está dividido em duas partes: Um modelo de referência, que fornece uma definição formal das entidades do modelo e seus relacionamentos, e uma especificação de necessidades, que define operações administrativas, funções de revisão e funções de sistema que podem ser utilizadas, respectivamente, para a criação e manutenção dos elementos e relações do modelo, para a execução de consultas administrativas e para criar e gerenciar sessões de usuários.

3.3.1. O Modelo de Referência

O objetivo central deste modelo de referência é 1) definir uma nomenclatura contendo termos para serem utilizados em aplicações RBAC e 2) especificar as características e funcionalidades do modelo. Este modelo de referência divide o RBAC em 4 componentes: a) *Core* RBAC, o qual define as entidades básicas do modelo e seus relacionamentos; b) *Hierarchical* RBAC, o qual define as especificações e relacionamentos para dar suporte à hierarquia entre papéis; c) *Static Separation of Duty Relations*, e d) *Dynamic Separation of Duty Relations* os quais definem os relacionamentos de separação de tarefas para assegurar que indivíduos não assumam papéis conflitantes de forma estática e dinâmica, garantindo as políticas de contrle de acesso das organizações. Os componentes c) e d) compõem o *Constrained RBAC.*

3.3.1.1. O *Core* RBAC

A Figura 3.5 apresenta o *Core* RBAC. Esta figura mostra os cinco elementos básicos do RBAC: usuários (*USERS*), papéis(*ROLES*), objetos(*OBS*),

operações(*OPS*) e permissões(*PRMS)*. Um papel é uma função desempenhada dentro do contexto de uma organização. Uma permissão é uma autorização para realizar uma operação em um objeto controlado pelo RBAC. Estas operações dependem do tipo de sistema no qual o controle de acesso está sendo implementado. Procedimentos de leitura, atualização e inclusão de registros em um tabela de um banco de dados são exemplos de operações.

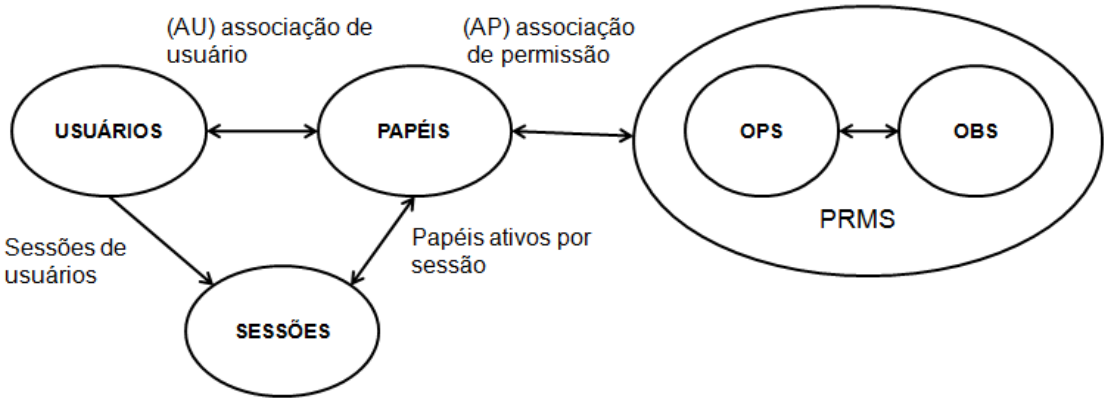

Figura 3. 5 - O *Core* RBAC
Fonte: Ferraiolo (2003)

Um objeto é uma entidade que contém ou recebe informações, e o acesso a essas informações precisa ser controlado. Arquivos, diretórios, aplicações, tabelas de um banco de dados e serviços são exemplos de objetos passíveis de serem controlados pelo RBAC.

A Figura 3.5 mostra dois relacionamentos importantes: A associação entre usuários e papéis (AU) e associação entre permissões e papéis (AP). Este relacionamento é de muitos-para-muitos, indicando que um usuário pode ser associado a vários papéis e um papel pode ser associado a vários usuários. De forma análoga, o mesmo comentário vale para a associação AP. Sessões são criadas quando da ativação de um usuário no sistema, e em seguida, papéis, pertencentes à lista de papéis autorizados a este usuário, são associados a ele.

3.3.1.2 O *Hierarchical* RBAC

Hierarchical RBAC define o relacionamento de herança entre papéis, uma característica idêntica à apresentada na seção 3.2, relacionada à herança de permissões Crampton (2003). A Figura 3.6 mostra a visão do RBAC hierárquico.

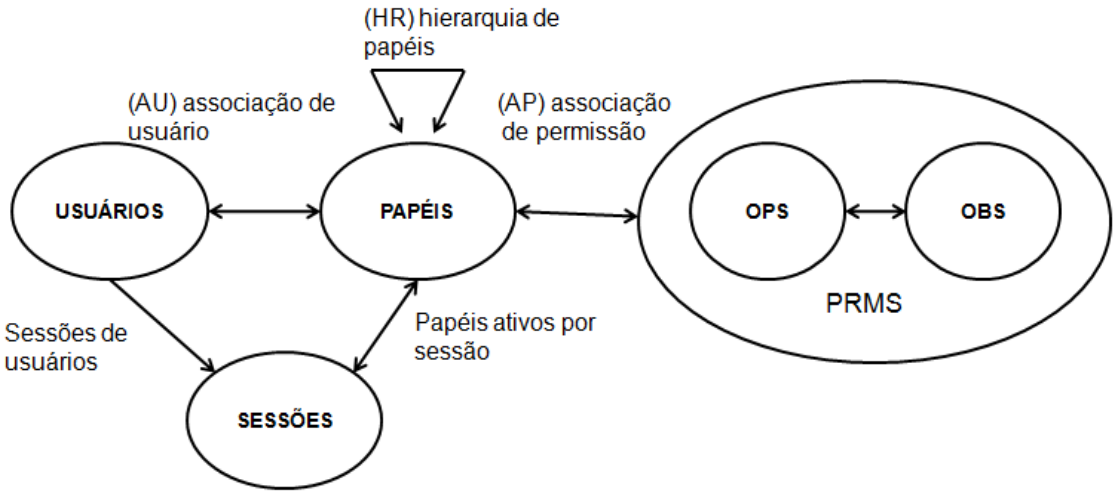

Figura 3. 6 - O *Hierarchical* RBAC
Fonte: Ferraiolo (2003)

A hierarquia entre papéis é uma forma de estruturar as funções e responsabilidades dentro de uma organização. Por convenção, as *roles* mais poderosas ou seniores, ou seja, *roles* com poucos usuários e mais permissões associadas, são colocadas no topo dos diagramas enquanto que as *roles* menos poderosas ou juniores são colocadas na parte inferior, conforme mostra a Figura 3.7. Na figura a *role* menos poderosa ou junior é a *role* **Enfermeira.** A *role* **Médico** é sênior com relação a *role* **Enfermeira** e portanto herda todas as permissões desta.

A *role* **Médico** pode ter mais permissões além daquelas herdadas da role **Enfermeira**. A herança de permissões é transitiva, então, a *role* **Clínico Geral** herda as permissões das *roles* **Médico** e **Enfermeira**.

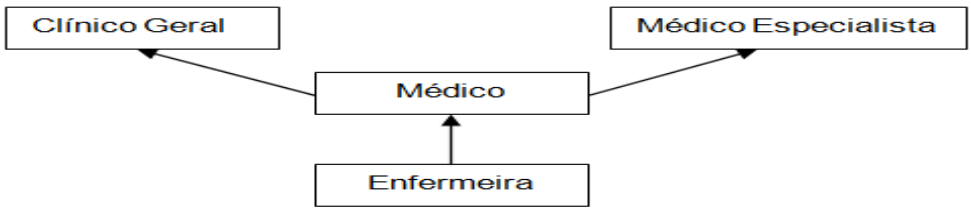

Figura 3. 7 - Exemplo de hierarquia de papéis
Fonte: Elaborado pelo autor

3.3.1.3. *Static Separation of Duty Relations(SSD)*

A separação estática de tarefas é uma forma de resolver o conflito entre as atividades desempenhadas dentro de uma organização. Neste caso, se um usuário está incluído na lista de membros de um papel A e há um relacionamento de *SSD* entre o papel A e um papel B, este usuário não poderá ser incluído na lista de membros do papel B. Para dar maior flexibilidade à construção deste relacionamento, Sandhu (2000) define a relação *SSD* com dois argumentos (*role set*, n), no qual o primeiro indica um conjunto de papéis (dois ou mais papéis), e o segundo informa a cardinalidade do conjunto. Por exemplo, para obrigar um usuário a assumir o papel 'r1' ou 'r2' deve ser definido um conjunto {r1, r2} com cardinalidade igual a 1 (o usuário pode exclusivamente assumir um papel do conjunto). A presença do relacionamento de *SSD* é ilustrada na Figura 3.8. A separação é estática porque ela está sempre ativa no sistema, agindo na relação usuário – papel(UA): se um usuário é autorizado como membro de um papel, ele é proibido de ser membro de um papel conflitante. Este relacionamento de *SSD* também deve ser respeitado pelo *Hierarchical* RBAC.

Por exemplo, seja um cenário contendo três papéis: A, B e C. Supondo que exista um conflito por *SSD* entre os papéis B e C e que o papel A herda o papel B, então A e C também possuem um conflito por *SSD* entre si.

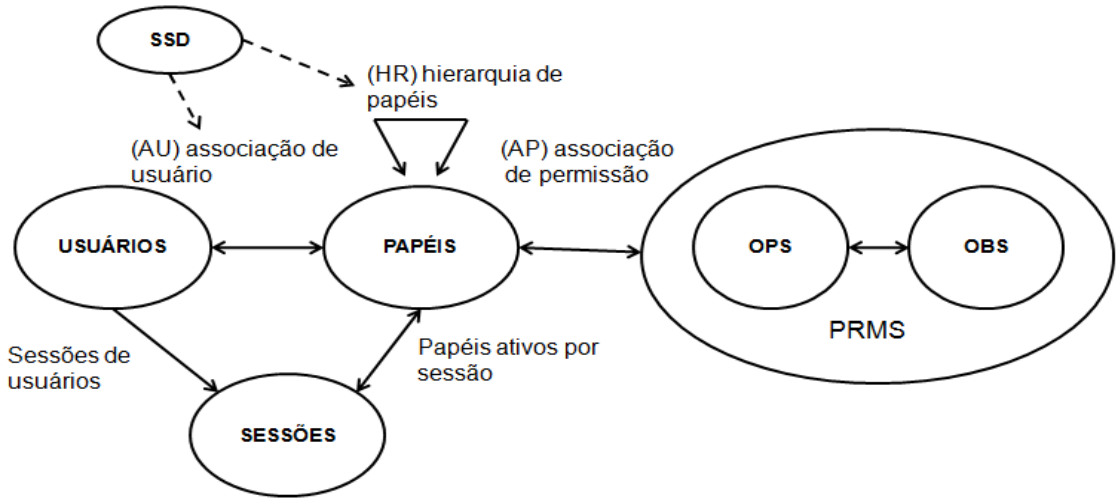

Figura 3.8 - O relacionamento SSD (Static Separation of Duty)
Fonte: Ferraiolo (2003)

3.3.1.4. *Dynamic Separation of Duty Relations(DSD)*

De forma semelhante ao *SSD*, a separação dinâmica de tarefas, ou *DSD*, também é utilizado para a limitação de permissões. No entanto, de forma menos restritiva, o *DSD* é criado no contexto dinâmico da ativação da sessão do usuário. Neste caso, se um usuário está incluído na lista de membros de um papel A e há um relacionamento de *DSD* entre este papel e um papel B, o usuário também pode estar incluído na lista de membros do papel B.Entretanto, durante uma sessão de usuário, ele não poderá assumir simultaneamente os dois papéis. O relacionamento de *DSD* é utilizado quando os papéis não criam conflito quando agem de forma independente, mas apenas simultaneamente. A Figura 3.9 ilustra esta situação. Também, de forma similar ao relacionamento de *SSD*, os relacionamentos de *DSD* são especificados com dois argumentos, o primeiro indicando o conjunto de papéis e o segundo indicando a cardinalidade do conjunto.

Por exemplo, um usuário não pode pertencer ao mesmo tempo as *roles* gerente de compras (que toma a decisão de realizar uma compra), e o gerente financeiro (que passa o cheque da compra), já que isso poderia abrir espaço para fraudes em que ele indicaria uma compra fraudulenta e autorizaria a sua

fatura por conta própria. Neste caso, uma mesma pessoa pode exercer os dois papéis, mas não ao mesmo tempo: estar ativo no sistema como gerente de compras e gerente financeiro permitira retirar dinheiro sem uma inspeção de terceiros. Com a separação dinâmica de deveres esse tipo de situação é tratada. O usuário seria obrigado a escolher um dos dois papéis para estar ativo no sistema, não podendo utilizar ao mesmo tempo os direitos de ambos.

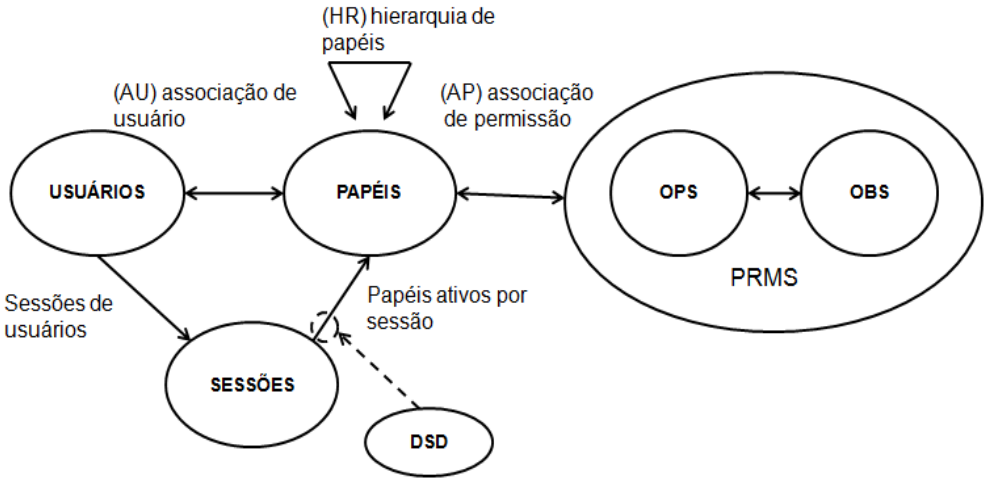

Figura 3. 9 - O relacionamento *DSD(Dynamic Separation of Duty)*
Fonte: Ferraiolo (2003)

3.3.2 A Especificação Funcional do Modelo RBAC-NIST

O modelo de referência anterior definiu uma série de entidades, componentes e relacionamentos. Para a criação e manutenção destes elementos do modelo RBAC, Ferraiolo (2001) propõe uma especificação de funções que fornecem a semântica de operação do modelo. São três as categorias de funções:

· Funções Administrativas – Funções de criação e manutenção dos elementos e relacionamentos para a construção dos vários modelos RBAC;

· Funções do Sistema – Funções de suporte necessárias durante a interação de um usuário com sistema (por exemplo, uma sessão de usuário);

· Funções de Revisão – Funções de verificação dos resultados das ações realizadas pelas funções administrativas;

A Tabela 3.5 resume estas especificações. As funções administrativas como *AddUser* e *AddRole* possibilitam, respectivamente, a inclusão de usuários e papéis no sistema.

Tabela 3.5 - Especificações funcionais do RBAC

ESPECIFICAÇÕES FUNCIONAIS DO RBAC				
	CORE RBAC	***HIERARCHICAL RBAC***	***Static Separation of Duty***	***Dynamic Separation of Duty***
FUNÇÕES ADMINISTRATIVAS	*AddUser* *DeleteUser* *AddRole* *DeleteRole* *AssignUser* *DeassignUser* *GrantPermission* *RevokePermission*	*AddInheritance* *DeleteInheritance* *AddAscendant* *AddDescendant* + *CORE RBAC*	*CreateSSDSet* *DeleteSSDSet* *AddSSDRoleMember* *DeleteSSDRoleMember* *SetSSDCardinality* + *CORE RBAC*	*CreateDSDSet* *DeleteDSDSet* *AddDSDRoleMember* *DeleteDSDRoleMember* *SetDSDCardinality* + *CORE RBAC*
FUNÇÕES DO SISTEMA	*CreateSession* *AddActiveRole* *DropActiveRole* *CheckAccess*	*CreateSession* *AddActiveRole* *DropActiveRole* *CheckAccess*	*CreateSession* *AddActiveRole* *DropActiveRole* *CheckAccess*	*CreateSession* *AddActiveRole* *DropActiveRole* *CheckAccess*
FUNÇÕES DE REVISÃO	*AssignedUsers* *AssignedRoles* *RolePermissions* *UserPermissions* *SessionRoles* *SessionPermissions*	*AssignedUsers* *AssignedRoles* *RolePermissions* *UserPermissions* *SessionRoles* *SessionPermissions* *AuthorizedUsers* *AuthorizedRoles*	*SSDRoleSets* *SSDRoleSetRoles* *SSDRoleSetCardinality* + *CORE RBAC*	*DSDRoleSets* *DSDRoleSetRoles* *DSDRoleSetCardinality* + *CORE RBAC*

Fonte: Ferraiolo (2003)

- *AssignUser* é a função responsável pela associação *UA* mostrada no modelo de referência, ou seja, é por meio desta função que usuários são incluídos na lista de membros de papéis.

- *GrantPermission* realiza o relacionamento *PA* do mesmo modelo.

- *AddInheritance* é responsável pela criação de um relacionamento hierárquico entre dois papéis existentes.

- *AddDescendant* cria um novo papel e o insere como descendente de um papel já existente.

- *CreateSession* cria uma sessão de usuário, ativando o usuário com um conjunto *default* de papéis ou permitindo a opção de escolha entre os papéis possíveis de serem ativados.

- *AddActiveRole* possibilita que um papel se torne ativo em uma dada sessão já estabelecida.

- *CheckAccess* verifica se um determinado *subject* de uma sessão tem permissão para executar a operação solicitada em um objeto.

- *CreateSSDSet* e *CreateDSDSet* criam, respectivamente, novos conjuntos de relacionamentos SSD e DSD.

- *AddSSDRoleMember* insere um papel como membro de um conjunto SSD existente

- *SetSSDCardinality* especifica a cardinalidade da composição dos papéis do conjunto correspondente.

- *AssignedUsers* é uma função de revisão que retorna os usuários contidos na lista membros de um dado papel.

- *RolePermissions* retorna o conjunto de permissões autorizadas para um determinado papel.

Como pode ser observado na Tabela 3.5, há uma repetição das funções ao longo dos diversos modelos do RBAC. Na realidade, as funções operam de forma idêntica, entretanto, faz-se necessário o suporte à nova funcionalidade introduzida pelo modelo correspondente.

3.4 Conclusões

Este capítulo apresentou o modelo de controle de acesso baseado em papéis que permite simplificar a administração da política de segurança facilitando a definição de políticas flexíveis e centralizadas. O RBAC agregou as melhores características dos modelos clássicos (discricionário e mandatório) no sentido de ser flexível como o primeiro e centralizado como o segundo Gallaher (2002).

O RBAC começou a ser formalizado em 1992 Ferraiolo (1992), apesar de ser um conceito da década de 70. O primeiro passo na direção de uma padronização foi dado pelo NIST (*National Institute of Standards and Technology*) com a proposta de um modelo unificado RBAC-NIST apresentado na seção 3.3.

Na construção do protótipo da ferramenta de configuração e gerência de papéis(*roles*) serão implementadas as regras de definição do modelo RBAC referentes ao *core* do RABC, a hierarquia de papéis e a separação estática de deveres.

4 PROJETO E IMPLEMENTAÇÃO

4.1 Introdução

A maioria dos sistemas de bancos de dados não implementa, na sua totalidade, as funcionalidades definidas no padrão ANSI do RBAC. Entre as funcionalidades menos implementadas estão as do modelo de restrições do RBAC, que inclui a separação estática e dinâmica de deveres. Poucos fabricantes de SGBD implementam estas restrições. Por exemplo, em um banco de dados Oracle 10g, um papel X pode ser atribuído a um usuário U, mesmo que este usuário já esteja associado a um papel Y, incompatível com X e uma vez estando associado a ambos os papéis o usuário pode colocar em risco a segurança da informação. O SGBD permitirá a realização da operação sem qualquer mensagem de alerta.

A Tabela 4.1 mostra uma comparação de funcionalidades RBAC de três SGBDs :

- *Informix Dynamic Server version 9.3*
- *Oracle Database 10g Release 2 (10.2)*
- *Sybase Adaptive Server version 15.0*

Tabela 4.1 - Comparação RBAC ente SGBDs

Item	Característica	Informix	Sybase	Oracle
1	Vários papéis ativos para uma sessão de usuário	Não	Sim	Sim
2	Hierarquia de papéis	Sim	Sim	Sim
3	Separação estática de papéis	Não	Sim	Não
4	Separação dinâmica de papéis	Sim [1]	Sim	Não

Fonte: Chandramouli (1998)

[1] No item 4 da tabela 4.1, o Informix suporta a separação dinâmica de papéis como um efeito colateral do item 1, pelo fato deste não poder ativar mais de um papel por sessão.

A proposta deste trabalho é criar uma ferramenta que possa ser utilizada com qualquer sistema de banco de dados e que implemente o *core* do RABC, a hierarquia entre papéis e a separação estática de deveres, também chamada de separação estática entre papéis, uma vez que estas funcionalidades, existentes no padrão ANSI do RBAC, foram estabelecidas em consenso entre pesquisadores do meio acadêmico e empresas comerciais, para promover o uso e o desenvolvimento da tecnologia de controle de acesso baseado em papéis incluindo as funcionalidades mais importantes e mais utilizadas.

Atualmente vários fabricantes de SGBDs implementam um subconjunto de funcionalidades do RBAC. Assim, em muitas situações a ferramenta irá extender as funcionalidades do modelo RBAC já existentes no banco de dados. Dentre as funcionalidades mais implementadas pelos bancos de dados estão o *core* do RBAC e a hierarquia entre papéis.

O protótipo da ferramenta, inicialmente, tem como objetivo principal implementar a separação estática entre papéis do modelo RBAC e evitar que ocorram associações de papéis que coloquem em risco a segurança da informação.

4.2 Especificação e Análise dos requisitos

O protótipo da ferramenta FCGP (Ferramenta de Configuração e Gerência de Papéis), que tem como usuário final a pessoa que exerce a função de DBA ou *Security Officer* dentro da organização, tem como objetivo implementar o *core* do RBAC, a hierarquia entre papéis e a separação estática entre papéis por meio da realização das seguintes tarefas:

- Criar e remover usuários
- Criar e remover papéis
- Conceder e revogar permissões aos papéis
- Conceder e revogar papéis aos usuários

- Conceder e revogar papéis a outros papéis
- Criar e remover restrições estáticas entre papéis
- Listar permissões de um determinado papel
- Listar papéis mutuamente exclusivos
- Listar hierarquia de papéis
- Listar papéis que possuem determinada permissão

Na criação de papéis, para cada tipo de organização onde a ferramenta for utilizada, será necessário um levantamento das funções e cargos dentro da organização e das permissões necessárias para a execução do cargo ou função. Com estas informações os papéis serão criados com as permissões necessárias correspondentes. Por exemplo, no caso de um hospital, pode-se definir os cargos ou funções de enfermeira e médico. A atualização de um campo de uma tabela que contenha os tipos sanguíneos de pacientes seria uma permissão necessária para a execução do cargo de enfermeira e a atualização de um campo de uma tabela que contenha o diagnóstico do estado de saúde do paciente seria uma permissão necessária para a execução do cargo de médico. Assim, dada uma tabela que armazena informações de pacientes, chamada **PACIENTE**, uma coluna chamada **TIPO_SANGUINEO** e uma coluna chamada **DIAGNOSTICO**, pode-se criar os papéis **Medico** e **Enfermeira**, com as permissões respectivas de **UPDATE** na coluna **DIAGNOSTICO** da tabela **PACIENTE** para o papel **Medico** e **UPDATE** na coluna **TIPO_SANGUINEO** da tabela **PACIENTE** para o papel **Enfermeira**.

Na tarefa de conceder e revogar permissões aos papéis, pode-se considerar as operações SELECT, INSERT, DELETE e UPDATE sobre objetos do tipo tabela, visão, coluna e índice, entretanto neste protótipo, por simplicidade, são considerados apenas objetos do tipo tabela.

Na associação de papéis aos usuários é feita uma verificação, no momento da associação, para saber se este papel não é mutuamente exclusivo com outro papel já associado a este usuário.

A hierarquia de papéis do modelo RBAC é implementada por meio da associação de papéis a outros papéis, na qual por exemplo, se um papel B é associado a um papel A, então A, assume todas as permissões do papel B.

As restrições de relacionamentos entre papéis também são definidas de acordo com o tipo de organização onde a ferramenta estiver sendo utilizada. Escolhe-se inicialmente um papel dentre uma lista de todos os papéis existentes e em seguida todos os papéis que são mutuamente exclusivos a este, criando-se assim o relacionamento de separação estática entre eles.

A ferramenta possui ainda a opção de listar as permissões associadas a um determinado papel, ou seja, selecionado um papel A, pode-se listar todas as operações (*SELECT, INSERT,DELETE e UPDATE*) sobre objetos (**tabelas**) que são permitidas aos usuários que estiverem associados a este papel. Outra opção importante, que serve para auxiliar na definição dos papéis mutuamente exclusivos é a de listar os papéis que possuem uma determinada permissão, ou seja, pode-se identificar quais papéis possuem a permissão para executar uma operação de *UPDATE* em uma determinada tabela. A Figura 4.1 mostra os casos de uso da ferramenta.

O objetivo do caso de uso principal **Definir Separação Estática** é cadastrar os papéis que apresentam conflitos de interesse, como por exemplo, um papel com autorização de solicitação de reembolso de despesas e outro com autorização de aprová-las. Uma organização pode proibir um mesmo usuário de estar associado a ambos os papéis. A ferramenta gera uma mensagem de aviso e impede o DBA ou S*ecurity Officer* de executar uma associação deste tipo.

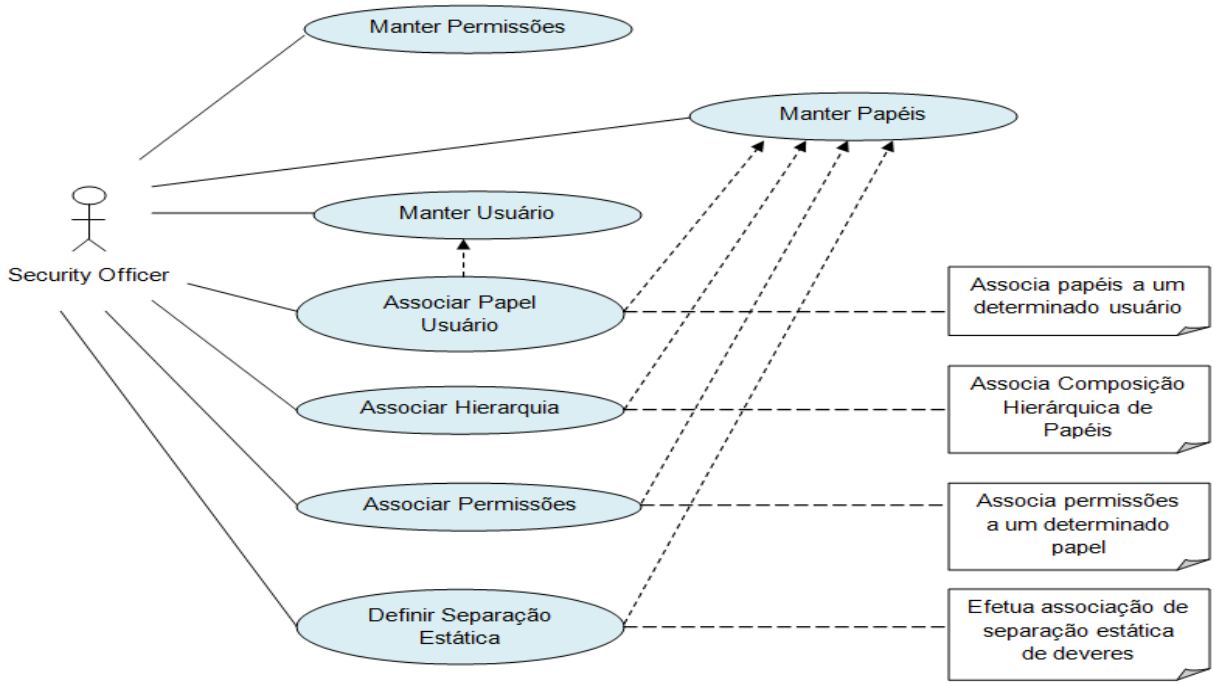

Figura 4.1 - Diagrama de Casos de Uso
Fonte: Elaborado pelo autor

4.3 Projeto

A FCGP armazena as informações referentes a usuários, papéis, associações entre permissões e papéis, associações entre usuários e papéis, associações entre papéis e papéis e regras de separação estática entre papéis, fornecidas pelo DBA ou *Security Officer* por meio de sua interface gráfica, em tabelas próprias, chamadas a partir de agora de tabelas do repositório. A FCGP utiliza estas informações e um conjunto de algoritmos próprios para implementar o *core* do RBAC, a hierarquia entre papéis e a separação estática entre papéis.

A Figura 4.2 mostra a atuação da FCGP no contexto geral do sistema como ferramenta de controle de acesso.

A FCGP propaga para o dicionário de dados do banco de dados apenas as informações contidas nas tabelas do repositório referentes ao *core* do RBAC e a hierarquia entre papéis.

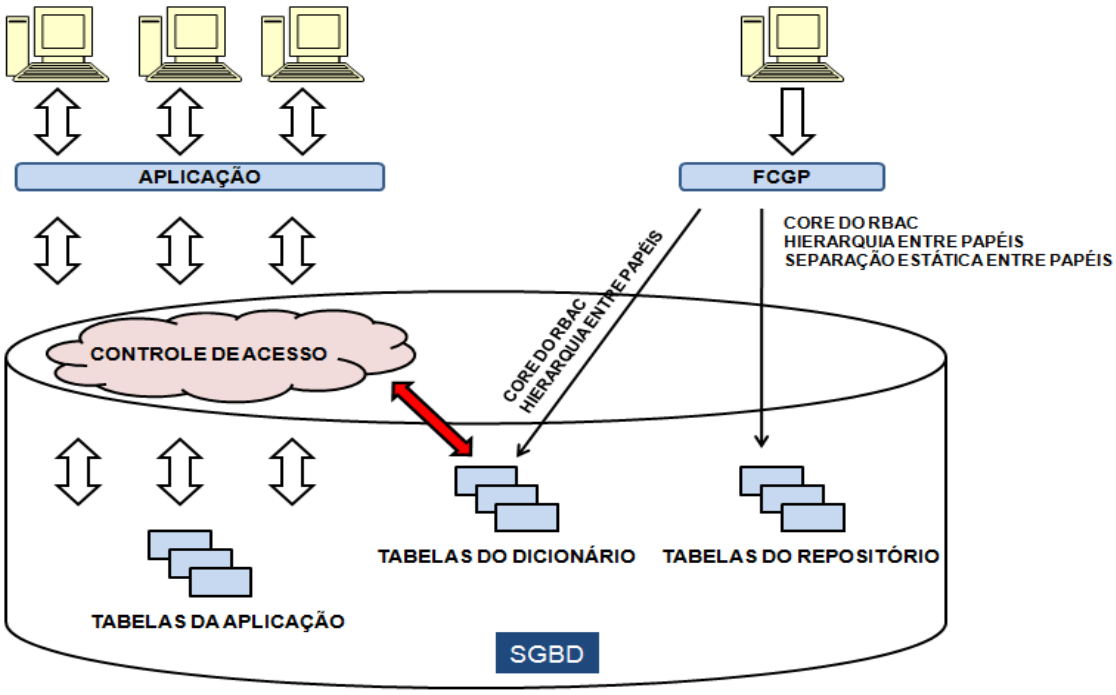

Figura 4. 2 - Funcionamento da FCGP

Fonte: Elaborado pelo autor

As informações sobre as regras de separação estática entre papéis não são propagadas porque a maioria dos bancos de dados não implementa a separação estática de deveres e portanto não existem tabelas ou visões no seu dicionário de dados para armazenar este tipo de informação. Por exemplo, quando o DBA ou *Security Officer*, utilizando a interface gráfica da FCGP, cria um papel P, o comando *CREATE ROLE* P é executado, por meio da interface JDBC, no banco de dados, armazenando o papel P nas tabelas do dicionário de dados daquele banco. Em seguida, a FCGP executa um comando *INSERT* que insere esta mesma informação em uma das tabelas do repositório.

Esta propagação de informações para o banco de dados ocorre utilizando-se os comandos do padrão SQL-99 *CREATE/DROP USER*, *CREATE/DROP ROLE* e *GRANT/REVOKE*. Embora os SGBDs de diversos fabricantes possuam extensões para os comandos *CREATE/DROP USER*,

CREATE/DROP ROLE e *GRANT/REVOKE*, a ferramenta utiliza apenas as opções definidas pelo padrão ANSI.

O motivo da propagação destas informações para o banco de dados é que, embora a ferramenta, usando seus algoritmos e as informações contidas nas tabelas do repositório, implemente as funcionalidades do RBAC, o controle de acesso das aplicações usuárias a suas tabelas continua sendo feito pelo próprio banco de dados. Quando uma aplicação usuária faz uma solicitação de acesso a alguma tabela do banco de dados, este verifica em seu dicionário de dados para saber se o usuário esta associado a algum papel que autorize o seu acesso.

Uma consequência desta propagação de informações é que se a FCGP é utilizada na definição da estrutura de controle de acesso de uma organização, o banco de dados gerenciado por ela, reflete a mesma estrutura com relação ao *core* do RBAC e a hierarquia entre papéis presente nas tabelas do repositório. Com relação as regras de separação estática entre papéis, embora elas não sejam propagadas e armazenadas no dicionário de dados do banco de dados, se elas forem criadas pelo DBA ou *Security Officer*, utilizando a FCGP e portanto armazenadas em uma das tabelas do repositório, nenhum usuário propagado para o dicionário de dados do banco de dados estará associado aos papéis que foram definidos como mutuamente exclusivos, uma vez que a FCGP impedirá o DBA ou *Security Officer* de realizar tal associação.

O objetivo de manter-se não apenas a informação sobre a separação estática entre papéis, mas também as informações sobre usuários, papéis, associações entre permissões e papéis, associações entre usuários e papéis e associações entre papéis e outros papéis em tabelas do repositório é de ter-se a opção futura da FCGP implementar o controle de acesso utilizando estas informações e não as informações armazenadas no dicionário de dados.

Neste contexto existem três aspectos importantes no projeto: o modelo relacional das tabelas do repositório, os algoritmos e a interface da FCGP.

4.3.1 Modelo Relacional da FCGP

As tabelas do repositório tem a função de armazenar as informações referentes a usuários, papéis, associações entre permissões e papéis, associações entre usuários e papéis, associações entre papéis e papéis e regras de separação estática entre papéis. A Figura 4.3 mostra o modelo relacional utilizado pela ferramenta.

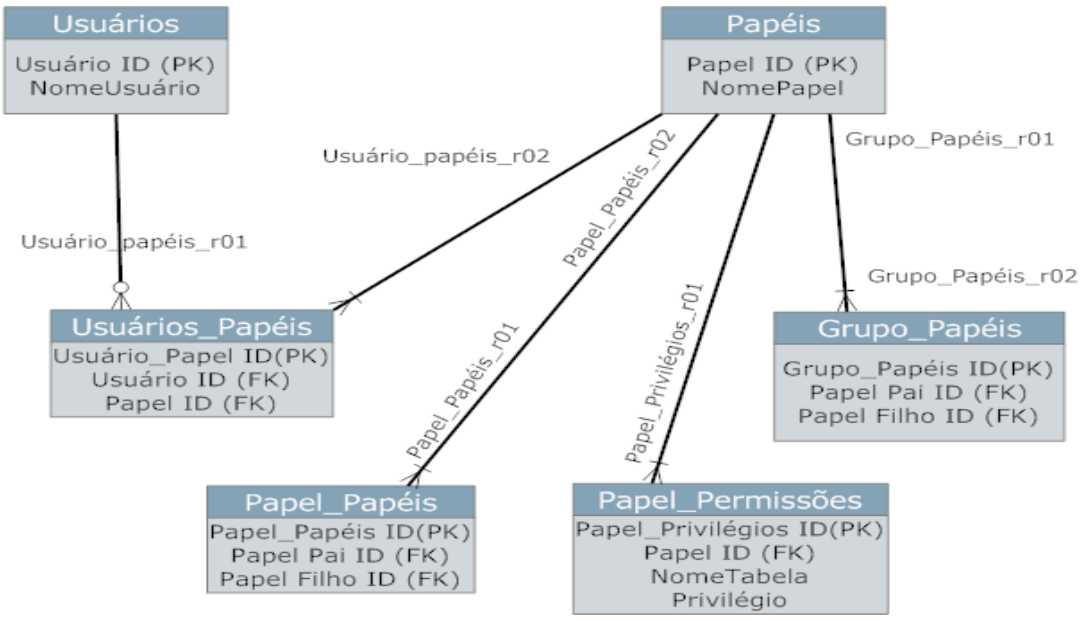

Figura 4. 3 - Modelo Relacional da FCGP
Fonte: Elaborado pelo autor

A Tabela 4.2 mostra uma descrição de cada uma das tabelas do repositório. Neste protótipo as tabelas do repositório são criadas no mesmo banco de dados que está sendo gerenciado pela ferramenta.

Todas as informações contidas nestas tabelas com exceção das regras de separação estática entre papéis (tabela **Papel_Papéis**) são propagadas para o banco de dados e armazenadas no seu dicionário de dados.

Tabela 4.2 - Descrição das tabelas da FCGP

Tabela	Descrição
Usuários	Armazena os usuários criados pela ferramenta
Papéis	Armazena os papéis criados pela ferramenta
Usuários_Papéis	Armazena os papéis de cada usuário
Grupo_Papéis	Armazena a hierarquia entre os papéis
Papel_Papéis	Armazena as restrições entre papéis
Papel_Permissões	Armazena as permissões de cada papel

Fonte: Elaborado pelo autor

4.3.2 Algoritmos da FCGP

Os algoritmos utilizados pela FCGP são divididos de acordo com a funcionalidade que implementam em algoritmos do *core* do RBAC, da hierarquia entre papéis e da separação estática entre papéis. Estes algoritmos utilizam as informações que estão armazenadas nas tabelas do repositório e que foram fornecidas pelo DBA ou *Security Officer* por meio da interface gráfica da FCGP.

Os algoritmos de implementação do *core* do RBAC são os mais simples porque além dos comandos SQL de CREATE/DROP e GRANT/REVOKE propagados para o banco de dados apenas operações de *INSERT* e *DELETE* nas tabelas do repositório são executadas. Os algoritmos são os seguintes:

- **Cria e remove usuário**: Executa os comandos SQL *CREATE* ou *DROP USER* no dicionario de dados e os comandos *INSERT* ou *DELETE* na tabela do repositório chamada **Usuários**.
- **Cria e remove papel**: Executa os comandos SQL *CREATE* ou *DROP ROLE* no dicionário de dados e os comandos *INSERT* ou *DELETE* na tabela do repositório chamada **Papéis**.

- **Associa e desassocia permissões aos papéis**: Executa os comandos SQL *GRANT* ou *REVOKE* no dicionário de dados e os comandos *INSERT* ou *DELETE* na tabela do repositório chamada **Papel_Permissões**.
- **Associa e desassocia usuários aos papéis**: Executa os comandos SQL *GRANT* ou *REVOKE* no dicionário de dados e os comandos *INSERT* ou *DELETE* na tabela do repositório chamada **Usuários_Papéis**.

Os algoritmos que implementam a hierarquia entre papéis são os seguintes:

- **Cria hierarquia entre papéis**: Um dos algoritmos da separação estática entre papéis, o que **Verifica se existe regra de separação estática entre papéis**, será utilizado aqui e descrito posteriormente. Seleciona-se inicialmente um papel X e em seguida, dentre uma lista de papéis disponíveis, escolhe-se o papel cujas permissões serão associadas a X, por exemplo o papel Y. Verifica-se a existência de separação estática entre X e Y. Caso não exista, um comando SQL *GRANT* Y *TO* X é propagado para o dicionário de dados e um *INSERT* na tabela **Grupo_Papéis** com os valores X e Y é feito. Caso exista, uma mensagem de erro é retornada e nada é feito.
- **Remove hierarquia entre papéis**: Seleciona-se inicialmente um papel X e em seguida, dentre uma lista de papéis hierarquicamente associados a ele, escolhe-se o papel cujas permissões serão desassociadas de X, por exemplo o papel Y. Um comando SQL *REVOKE* Y *FROM* X é propagado para o banco de dados e um *DELETE* na tabela **Grupo_Papéis** removendo a associação entre X e Y é feito.
- **Lista papéis hierarquicamente associados a um papel**: Dado um papel X, este algoritmo retorna os papéis hierarquicamente associados a X consultando a tabela do repositório **Grupo_Papéis**. Esta lista é utilizada pelo algoritmo que **Verifica se existe regra de separação estática entre papéis**, uma vez que quando não existe

uma separação estática direta entre os papéis na tabela **Papel_Papéis**, o algoritmo precisa identificar se existe uma separação estática entre os papéis hierarquicamente associados a eles.

Os algoritmos que implementam a separação estática entre papéis não propagam informações para o dicionário do banco de dados. São eles :

- **Cria regra de separação estática entre papéis**: Seleciona-se inicialmente um papel X e em seguida, dentre uma lista de papéis disponíveis, escolhe-se o papel que estará em separação estática com X, por exemplo o papel Y. Utiliza-se o algoritmo que **Verifica se existe regra de separação estática entre papéis**. Caso não exista, insere-se estes dois papéis na tabela **Papel_Papéis**. Caso exista, uma mensagem de erro é retornada e nada é feito.
- **Remove regra de separação estática entre papéis**: Seleciona-se inicialmente um papel X e em seguida, dentre uma lista de papéis que estão em separação estática com X, escolhe-se o papel que será removido da separação estática com X, por exemplo o papel Y. Remove-se da tabela **Papel_Papéis** associações entre X e Y.
- **Verifica se existe regra de separação estática entre papéis**: Dado um papel X e um papel Y, este algoritmo verifica se existe uma regra de separação estática entre X e Y. Para isso, além de verificar se existe uma regra de separação estática direta entre X e Y, o algoritmo verifica também se existe uma regra de separação estática entre X e os demais papéis associados hierarquicamente a Y e vice versa, ou seja, se existe uma regra de separação estática entre Y e os demais papéis associados hierarquicamente a X. A existência de regras de separação estática entre papéis associados a X e Y implica que existe uma separação estática entre X e Y. Estas listas de papéis associados a X e Y são obtidas utilizando-se o algoritmo **Lista papéis hierarquicamente associados a um papel**.

4.3.3 Interface da FCGP

A interface gráfica da FCGP tem a função de possibilitar ao DBA ou *Security Officer* ativar todas as funcionalidades listadas no item 4.2. Após o login no banco de dados, uma tela com as opções **Papéis, Usuários** e **Permissões** é apresentada. Ao selecionar-se a opção **Papéis**, as informações sobre permissões associadas aos papéis, hierarquia entre papéis e papéis que se encontram em separação estática são visualizadas conforme mostra a Figura 4.4.

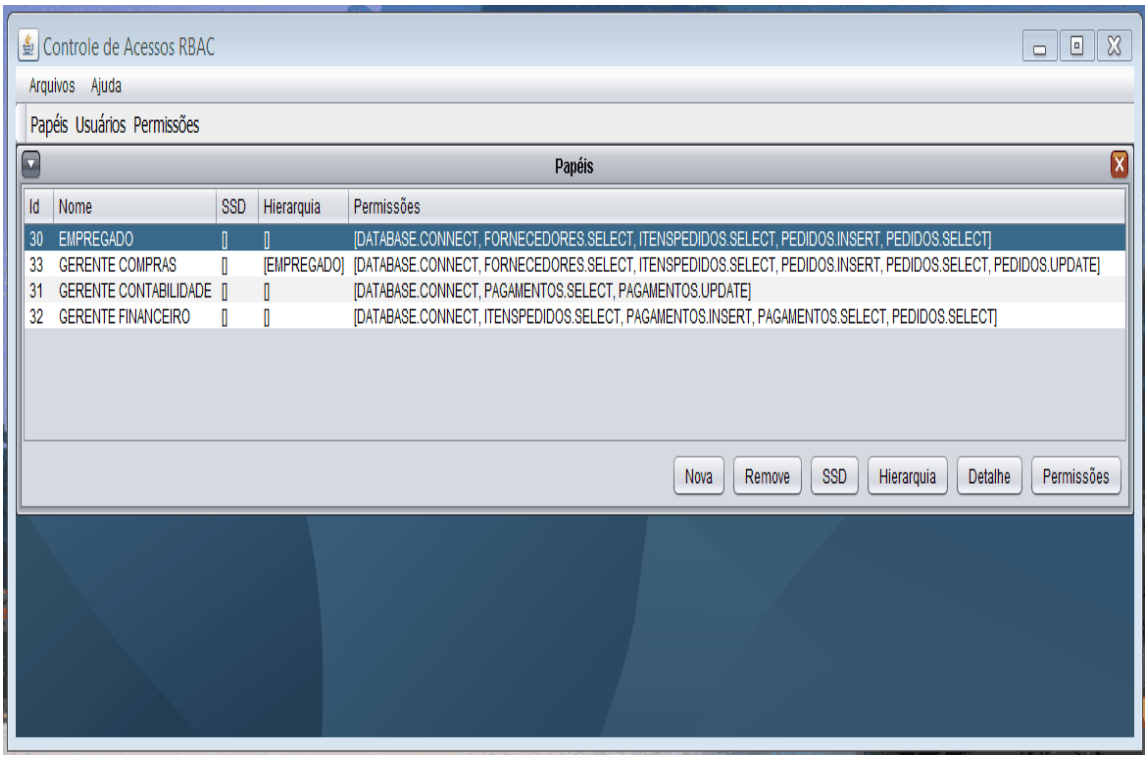

Figura 4. 4 - Tela da FCGP com informações sobre papéis
Fonte: Elaborado pelo autor

A Figura 4.5 mostra como é feita, por meio da ferramenta, a inclusão e exclusão de restrições entre papéis. O DBA ou *Security Officer* , após selecionar a opção **Papéis**, seleciona o papel sobre o qual irá criar ou remover uma regra de separação de deveres, no exemplo, o papel **Gerente Financeiro**, em seguida, ao selecionar-se o botão SSD, é apresentada uma tela com os papéis

disponíveis e com os papéis que já se encontram em separação estática com o papel **Gerente Financeiro**. O DBA ou *Security Officer* pode então selecionar os papéis que deseja adicionar ou remover do conjunto de papéis mutuamente exclusivos ao papel **Gerente Financeiro**.No exemplo, é selecionado e adicionado o papel **Gerente Contabilidade**. Esta informação fica armazenada na tabela da aplicação **Papel_Papéis**.

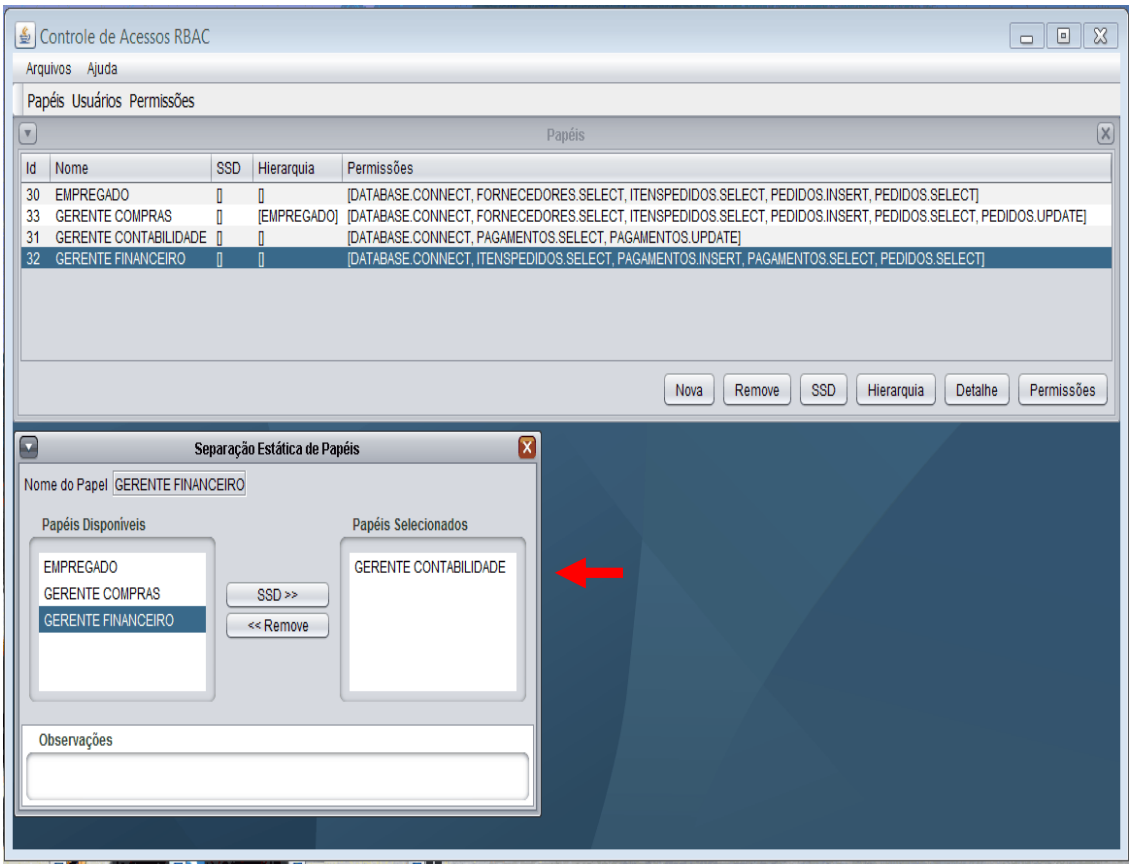

Figura 4. 5 - Inclusão de separação estática de deveres entre papéis

Fonte: Elaborado pelo autor

Uma vez que os usuários, papéis, associações entre permissões e papéis, associações entre usuários e papéis, associações entre papéis e outros papéis e a separação estática entre papéis estão definidos, a ferramenta, que tem como objetivo principal evitar a associação indevida de papéis conflitantes para um mesmo usuário, gera uma mensagem de alerta caso o DBA ou *Security Officer* tente executar uma associação indevida. A Figura 4.6 mostra um

exemplo desta situação quando tenta-se associar o papel **Gerente Financeiro** ao usuário **Jenner** que já está associado ao papel **Gerente Contabilidade**, uma vez que estes papéis foram definidos como mutuamente exclusivos.

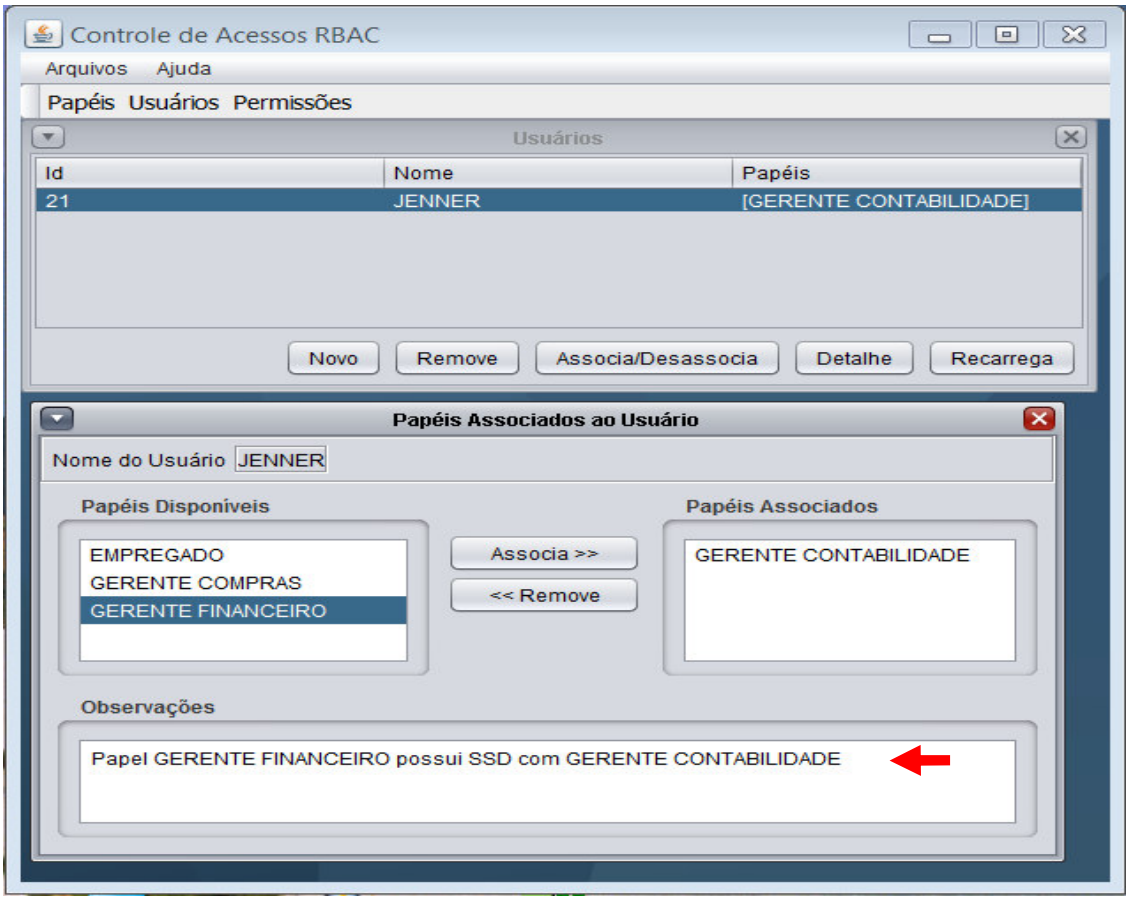

Figura 4.6 - Exemplo de tentativa de associação entre papéis mutuamente exclusivos
Fonte: Elaborado pelo autor

O DBA ou *Security Officer* seleciona a opção de **Usuários**, a tela que aparece em seguida mostra os usuários e os papéis associados a eles, em seguida seleciona o usuário **Jenner,** que tem o papel **Gerente Contabilidade** associado a ele**,** seleciona o botão **Associa/Desassocia** e a tela **Papéis Associados ao Usuário** mostra os papéis disponíveis e os já associados ao usuário. Quando o DBA ou *Security Officer* seleciona o papel **Gerente Financeiro** e seleciona o botão **Associa** a aplicação verifica na tabela

Papel_Papéis que estes papéis estão em separação estática de deveres e emite a mensagem **Papel Gerente Financeiro possui SSD com Gerente Contabilidade**, evitando assim que o usuário **Jenner** adquira permissões que coloquem em risco a segurança do negócio.

4.4 Aspectos de Implementação

O protótipo da ferramenta FCGP foi implementado utilizando-se a linguagem Java 2 e o ambiente de desenvolvimento do NetBeans 6.5. O SGBD utilizado nos testes foi o Oracle 10g e ele foi escolhido por não implementar a separação estática de deveres. A conexão com o SGBD foi feita por meio da API JDBC.

A FCGP atua na configuração e na gerência dos usuários, papéis, hierarquias e regras de separação estática entre eles, entretanto fica a cargo do banco de dados o cumprimento destas regras, ou seja, quando uma aplicação usuária tenta realizar alguma operação em uma tabela do banco de dados, este, com base nas informações contidas no seu dicionário de dados autoriza ou não o acesso. Com relação a separação estática entre papéis, uma vez que, utilizando a FCGP, foi criada uma regra de separação estática entre papéis e esta informação foi armazenada nas tabelas do repositório, a FCGP impede que estes papéis sejam associados a um mesmo usuário, impedindo assim que uma associação deste tipo seja propagada para o dicionário de dados por meio de um comando *GRANT*. Assim, os usuários criados pela FCGP e portanto existentes nas tabelas do repositório e propagados para dentro do dicionário de dados, não estarão associados a papéis mutuamente exclusivos, garantindo assim a separação estática de deveres. Este é o objetivo principal da FCGP, garantir que nenhum usuário dentro do banco de dados tenha sido associado a papéis definidos como mutuamente exclusivos por ela.

Apenas a FCGP deve ser utilizada na configuração e na gerência dos usuários, papéis, hierarquias e regras de separação estática entre papéis, devido a questões relacionadas a segurança e consistência das informações

armazenadas nas tabelas do repositório e daquelas armazenadas no dicionário de dados.

A principal questão relacionada a segurança e ao uso da FCGP é que caso o DBA ou *Security Officer* utilize uma outra ferramenta para se conectar ao banco de dados, como por exemplo o **SQL*PLUS** da Oracle, e por meio dela execute um comando SQL *GRANT* que faça uma associação de um papel a um usuário, não ocorrerá uma verificação nas tabelas do repositório se este papel está em separação estática com algum outro papel já associado ao usuário. Com isso, papéis que foram definidos pela FCGP como mutuamente exclusivos poderão ser associados a um mesmo usuário, caracterizando uma falha no controle de acesso.

Com relação a consistência das informações entre as tabelas do repositório e o dicionário de dados, caso o DBA ou *Security Officer* utilize uma outra ferramenta para se conectar ao banco de dados, como por exemplo o **SQL*PLUS** da Oracle, e por meio dela execute um comando SQL CREATE ROLE para criar um papel novo, a tabela do repositório que armazena as informações sobre papéis, chamada **Papéis,** não será atualizada com esta informação, deixando assim as tabelas do repositório e as tabelas do dicionário de dados inconsistentes.

A fim de tentar resolver os problemas relacionados a consistência e a segurança, a FCGP utiliza um recurso disponível em alguns SGBDs, conhecido como *triggers de DDL*.

4.4.1 *Triggers* de DDL (*Data Definition Language*)

Assim como os *triggers* de DML (*Data Manipulation Language*), os *triggers* de DDL são programas ou procedimentos armazenados que são executados em resposta a um evento. Entretanto, os *triggers* de DML são executados em resposta aos eventos de manipulação de dados *INSERT*, *UPDATE* e *DELETE* em tabelas ou visões e os *triggers* de DDL são disparados

em resposta aos eventos de definição de dados *CREATE*, *ALTER*, *DROP*, *GRANT*, *REVOKE* permitindo capturar qualquer evento que possa criar, alterar, excluir ou atribuir e revogar permissões em diferentes objetos de um banco de dados.

Os *triggers* de DDL tem um escopo maior do que os *triggers* de DML podendo ser definidos no nível de banco de dados ou de *schema*. Pode-se, por exemplo, utilizar um *trigger* de DDL para monitorar uma alteração de uma coluna de uma tabela, um comando de *DROP TABLE* ou ainda uma operação de *shutdown* em um banco de dados.

Os códigos fonte dos *triggers* de DML e de DDL são armazenados em tabelas do dicionário de dados de cada banco.

Entre os fabricantes de SGBDs que fornecem suporte para a criação de *triggers* de DDL estão a Oracle, com o Oracle Enterprise Server 10g, a Microsoft, com o SQLServer 2005 e a Sun MicroSystems, com o MySQL 5.1.

4.4.2 *Triggers* de DDL da FCGP

No momento da instalação da FCGP, para os SGBDs que possuem o recurso de *trigger* de DDL, a ferramenta, executando comandos *CREATE TRIGGER*, cria no dicionário de dados do banco de dados quatro *triggers* de DDL, um para o comando *GRANT*, um para o comando *REVOKE*, um para o comando *CREATE* e outro para o comando *DROP* . O *trigger* para o comando SQL *GRANT* é disparado sempre antes que algum comando *GRANT* seja executado e dependendo das associações utilizadas no comando, executa as seguintes tarefas:

- *GRANT* utilizado para associar permissões a papel: Neste caso o código do *trigger* é utilizado para manter a consistência das informações entre as tabelas do repositório e as tabelas do dicionário de dados, inserindo na tabela **Papel_Permissões** as novas permissões associadas a um determinado papel.

- *GRANT* utilizado para associar papéis a usuário: Neste caso o código do *trigger* verifica, consultando as tabelas do repositório, se já existe algum papel associado ao usuário e que esteja em separação estática com o papel que se deseja associar. Caso exista uma regra de separação estática entre os papéis, o *GRANT* não é executado e uma mensagem , avisando sobre a separação estática, é retornada. Caso não exista uma regra de separação estática entre o papéis, o *GRANT* é executado com sucesso, atualizando o dicionário de dados. A tabela do repositório **Usuários_Papéis** é atualizada inserindo-se os papéis associados a um determinado usuário. O objetivo principal deste *trigger* de DDL é evitar que um papel seja associado a um usuário por meio de alguma outra ferramenta que permita a execução do comando *GRANT* diretamente no banco de dados, que não a FCGP. Isto implicaria na existência de um usuário associado a papéis definidos pela FCGP como mutuamente exclusivos e portanto numa falha de segurança que colocaria em risco o negócio.
- *GRANT* utilizado para associar papéis a outro papel: Neste caso o código do *trigger* verifica, consultando as tabelas do repositório, se existe alguma regra de separação estática entre os papéis que se deseja associar e os papéis que receberão as associações. Caso exista tal regra, o comando *GRANT* não é realizado e uma mensagem avisando sobre a separação estática é retornada. Caso não exista tal regra, o *GRANT* é realizado com sucesso, as tabelas do dicionário de dados são atualizadas e o código do *trigger* , com o objetivo de manter a consistência das informações, insere na tabela do repositório resposável por armazenar as informações sobre hierarquia entre papéis, chamada **Grupo_Papéis** os papéis associados a um determinado papel.

A Figura 4.7 mostra o código do *trigger* de DDL para o comando SQL *GRANT* quando utilizado para associar usuários aos papéis.

```
TRIGGER SSD_GRANT BEFORE GRANT ON DATABASE
DECLARE
 PRAGMA AUTONOMOUS_TRANSACTION;
 v_grant_type      VARCHAR2 (30);
 v_num_grantees    BINARY_INTEGER;
 v_grantee_list    ora_name_list_t;
 v_num_privs       BINARY_INTEGER;
 v_priv_list       ora_name_list_t;
 plsql_block       VARCHAR2(500);
BEGIN
 plsql_block       := 'INSERT into USER_ROLES values(:USER,:ROLE);'
 v_grant_type      := ora_dict_obj_type;
 v_num_privs       := ora_privilege_list(v_priv_list);
 v_num_grantees    := ora_grantee (v_grantee_list);
 FOR counter1 IN 1..v_num_privs
 LOOP
    FOR counter2 IN 1..v_num_grantees
    LOOP
       CHECK_ROLE_SSD(v_priv_list (counter1), v_grantee_list(counter2));
       EXECUTE IMMEDIATE plsql_block USING v_grantee_list(counter2), v_priv_list (counter1);
       COMMIT;
    END LOOP;
 END LOOP;
END;
```

Figura 4. 7 - Exemplo de *trigger* de DDL (Data Definition Language) no Oracle
Fonte: Elaborado pelo autor

O algoritmo **Verifica se existe regra de separação estática entre papéis** é implementado pela procedure PL/SQL **CHECK_ROLE_SSD**.

No caso do comando *REVOKE*, o código do *trigger* associado a ele realiza as seguintes alterações :

- *REVOKE* utilizado para desassociar permissões de papel: Neste caso o código do trigger remove da tabela do repositório **Papel_Permissões** as permissões associadas a um determinado papel, mantendo a consistência entre esta tabela e o dicionário de dados.

- *REVOKE* utilizado para desassociar papéis de usuários : Neste caso o código do trigger remove da tabela do repositório **Usuários_Papéis** os papéis asociados ao usuário, mantendo a consistência entre esta tabela e o dicionário de dados.
- *REVOKE* utilizado para desassociar papéis de outro papel: Neste caso o código do trigger remove da tabela do repositório **Grupo_Papéis** os papéis associados a um determinado papel, mantendo a consistência entre esta tabela e o dicionário de dados.

Os *triggers* para os comandos SQL *CREATE* e *DROP* fazem inserções e exclusões das tabelas do repositório **Usuários** e **Papéis** respectivamente.

O objetivo principal destes *triggers* é evitar que ocorra uma associação de papéis enganosa ou fraudulenta e garantir a consistência das informações entre as tabelas do repositório e as tabelas do dicionário de dados.

Para os SGBDS que não implementam o recurso de *triggers* de DDL , apenas a FCGP poderá ser utilizada para a configuração e gerencia do *core* do RBAC, da hierarquia e da separação estática entre papéis a fim de garantir-se a segurança e consistência das informações.

4.5 Conclusões

Este capítulo especificou as funcionalidades da ferramenta FCGP, a criação de usuários e papéis, a associação de papéis a usuários, a associação de permissões a papéis, a associação de papéis a outros papéis e a criação de regras de separação estática entre papéis. O capítulo apresentou o modelo relacional da FCGP mostrando em quais tabelas do repositório as informações de usuários, papéis, permissões, hierarquia e restrições estáticas entre papéis são armazenadas e ainda quais os algoritmos são utilizados por ela para implementação do *core* do RBAC, da hierarquia entre papéis e da separação estática entre papéis. Foi visto ainda que a FCGP propaga apenas as informações do *core* do RBAC e da hierarquia entre papéis para as tabelas do

dicionário de dados e que as informações sobre as regras de separação estática entre papéis não são propagadas para as tabelas do dicionário, porque por não implementar a separação estática entre papéis, o SGBD não possui tabelas no seu dicionário para armazenar este tipo de informação . Entretanto mesmo não propagando estas informações, a FCGP evita que papéis mutuamente exclusivos sejam associados a um mesmo usuário, utilizando as informações das tabelas do seu repositório. Com isso nenhuma associação deste tipo será propagada para o dicionário do banco de dados e portanto nenhum usuário definido no dicionário de dados estará associado a papéis definidos pela FCGP como sendo mutuamente exclusivos.

A FCGP possui ainda a opção, para bancos de dados que suportem esta funcionalidade, da criação de *triggers* de DDL. O objetivo principal destes *triggers* é evitar que uma operação de *GRANT* executada por outra aplicação que não a FCGP possa ser realizada sem que uma consulta a tabela do repositório que armazena as restrições de relacionamentos entre papéis seja feita, permitindo uma associação indevida. Os demais *triggers* são criados para manter-se a consistência das informações entre as tabelas do repositório e o dicionário de dados.

Neste capítulo são apresentadas algumas telas de interface da FCGP e que mostram como as informações sobre papéis são apresentadas, como uma inclusão de separação estática entre papéis é feita e como uma tentativa de associação entre papéis mutuamente exclusivos é bloqueada.

5 TESTES DA FCGP

5.1 Introdução

Com o objetivo de testar e validar como a ferramenta pode ser utilizada para definir usuários, permissões, papéis, hierarquia de papéis e garantir a separação estática entre eles, um estudo de caso hipotético envolvendo a configuração de controle de acesso baseado em papéis em uma instituição financeira é apresentado. Primeiramente é apresentado o cenário no qual o controle de acesso é feito baseado no modelo discricionário. São apresentadas as aplicações envolvidas, os usuários, os cargos e as regras de operação do banco. Em seguida, inicia-se o processo de modelagem da política para o controle de acesso RBAC, definindo-se os papéis, as permissões, a hierarquia e a separação estática entre eles. Utilizando-se estas informações, são apresentadas as sequências de telas e ações necessárias para a implementação deste modelo por meio da FCGP. No final do capítulo tem-se um exemplo de um modelo econômico simples que pode ser usado para se ter uma idéia aproximada da economia resultante do uso de um controle de acesso baseado em papéis em relação a um modelo discricionário, e tem-se uma sugestão de trabalho futuro utilizando-se a FCGP e o RBAC.

5.2 Cenário com controle de acesso discricionário

Para este estudo de caso a instituição financeira será chamada de Banco Hipotético. O Banco usa uma variedade de aplicações para dar suporte ao seu negócio. O controle de acesso a estas aplicações é realizado de forma discricionária, onde cada aplicação mantém uma tabela própria para o controle dos direitos de acesso dos usuários. Portanto, para cada usuário, uma aplicação deve ter uma entrada nesta tabela de direitos de acesso com as operações que ele pode realizar durante sua sessão. A manutenção deste controle de acesso é sujeito a erro e custoso, pois os direitos de acesso são administrados

individualmente, com sua complexidade crescendo na proporção do crescimento do número de usuários e de aplicações no sistema.

A Tabela 5.1 mostra as duas aplicações analisadas neste estudo de caso e suas principais funcionalidades. Os usuários do sistema e seus cargos dentro do banco são mostrados na Tabela 5.2.

Tabela 5. 1 - Aplicações e funcionalidades do Banco Hipotético

Aplicação	Funcionalidades
Aplicação Financeira	Agendar TED - Agendar DOC Autorizar TED - Autorizar DOC Auditar Transações – Efetuar Pagamentos
Aplicação de Clientes	Registar nova ContaCorrente – Conceder Limite Auditar Transações

Fonte: Elaborado pelo autor

Tabela 5. 2 - Usuários e cargos do Banco Hipotético

Usuários	Cargos
Carlos e Ana	Atendente
Maria e Silvia	Caixa
Pedro	Supervisor e Atendente
Paulo	Supervisor
Antonio	Auditor
Sérgio	Funcionário

Fonte: Elaborado pelo autor

Segundo as regras de operação do banco, os empregados que ocupam a posição de **Atendente** podem apenas agendar transações TED (Transferência Eletrônica Disponível) e DOC (Documento de Crédito) na Aplicação Financeira e abrir contas por meio da Aplicação de Clientes; os empregados que ocupam o cargo de **Supervisor** podem conceder limite a uma conta corrente e ainda autorizar DOCs e TEDs; os empregados da posição **Caixa** podem realizar as operações conferidas a um **Atendente** e ainda efetuar pagamentos; os empregados associados ao cargo **Auditor** nunca poderão realizar outra operação senão aquelas relacionadas com os procedimentos de auditoria das transações efetuadas, como por exemplo, armazenar nas trilhas de auditoria a alteração de uma coluna contendo o valor do salário de um funcionário e o responsável pela alteração. Todas as operações devem ser realizadas no horário de expediente bancário, que vai das 10:00hs às 16:00hs, de segunda-feira à sexta-feira. Por questões de segurança, apenas os acessos provenientes da rede interna poderão executar as operações de auditoria.

Neste cenário, o administrador do sistema deve incluir, individualmente para cada usuário, todas as restrições impostas pelas regras do banco, o que é, de fato, uma atividade custosa e sujeita a erro. Um outro problema observado é que a responsabilidade pela realização das tarefas de controle de acesso fica a cargo da própria aplicação financeira, procedimento que não faz parte da atividade principal para a qual ela foi concebida.

5.3 Cenário com controle de acesso RBAC

Um papel pode ser visto como um conjunto de atividades e responsabilidades associadas a um determinado cargo ou função. Assim, ao invés de especificar um conjunto de acessos autorizados para cada usuário do sistema, as permissões são conferidas aos papéis. Por conseguinte, um usuário que exerce um papel pode realizar todos os acessos para os quais o papel está autorizado. Entretanto nem sempre tem-se uma relação direta entre a ocupação do indivíduo dentro da organização e o papel RBAC correspondente na hierarquia. Este processo, chamado *role engineering*, envolve uma análise mais

refinada das permissões associadas e dos níveis de responsabilidades das ocupações. Neste estudo de caso, entretanto, considera-se uma relação direta entre o cargo ocupado pelo indivíduo dentro da organização e o papel RBAC, assim são definidos os seguintes papéis : **Funcionário**, **Atendente**, **Caixa**, **Supervisor** e **Auditor**.

Uma permissão pode ser vista como a aprovação de uma determinada operação em um determinado objeto e embora tanto operação quanto objeto sejam conceitos dependentes de cada sistema, neste estudo de caso a FCGP utiliza apenas as operações SELECT,INSERT,DELETE e UPDATE sobre objetos do tipo **tabela**. A Tabela 5.3 apresenta as associações entre as funcionalidades das aplicações e os tipos de permissões necessárias para a execução desta funcionalidade.

Com relação a hierarquia, para o estudo de caso em questão, a análise das ocupações dos funcionários e de suas atribuições no banco leva à hierarquia de papéis mostrada na Figura 5.1.Na hierarquia de papéis proposta, o papel de nível inferior denominado **Funcionário** tem associado um conjunto de permissões de acesso básicas, comum a todo funcionário do banco, como por exemplo a permissão para se conectar ao sistema. Cada papel possui um conjunto de permissões próprias mais aquelas herdadas dos papéis de níveis inferiores na hierarquia.

Tabela 5. 3 - Funcionalidades e permissões do Banco Hipotético

Funcionalidades	Permissões
Agendar TED	Insert na tabela TED
Agendar DOC	Insert na tabela DOC
Autorizar TED	Select,Update na tabela TED
Autorizar DOC	Select,Update na tabela DOC
Efetuar Pagamentos	Selec,Update na tabela PAG
Registrar nova Conta Corrente	Insert na tabela CC
Conceder Limite	Select,Update na tabela CC
Auditar Transações	Select em TED,DOC,PAG,CC

Fonte: Elaborado pelo autor

As permissões associadas com o papel **Funcionário** são herdadas pelo papel **Atendente**. Aquelas associadas a **Atendente** são herdadas por **Caixa**.Os papéis **Auditor** e **Supervisor** herdam as permissões do papel **Funcionário** como mostra a Figura 5.1.

Figura 5. 1 - Hierarquia de papéis do Banco Hipotético
Fonte: Elaborado pelo autor

A Tabela 5.4 apresenta os direitos de acesso atribuídos a cada papel da hierarquia.

Tabela 5. 4 - Papéis e direitos de acesso

Papel	Aplicação Financeira	Aplicação de clientes
Atendente	Agendar TED Agendar DOC	Registrar nova conta corrente
Supervisor	Autorizar TED Autorizar DOC	Conceder Limite
Caixa	Efetuar Pagamentos Agendar TED Agendar DOC	Registrar nova conta corrente
Auditor	Auditar Transações	Auditar Transações
Funcionário	Conectar-se ao SGBD	Conectar-se ao SGBD

Fonte: Elaborado pelo autor

5.4 Separação estática entre papéis

A idéia principal da separação estática entre papéis é a de se dividir uma tarefa em funcionalidades de modo que nenhum papel tenha permissões suficientes para executar todas as funcionalidades. Assim, por exemplo, no caso do Banco Hipotético, tem-se que a tarefa de “Abrir conta corrente” foi dividida em duas funcionalidades, “Registrar nova conta corrente” e “Conceder limite”. Cada uma destas funcionalidades foi associada a um papel diferente, **Atendente** e **Supervisor**, respectivamente. Quando é definida uma regra de separação estática entre eles, impedindo-se que uma mesma pessoa seja associada a ambos, garante-se que duas ou mais pessoas sejam necessárias para se completar toda a tarefa de “Abrir conta corrente”. A necessidade de duas ou mais pessoas para realização de uma tarefa coibi ações fraudulentas uma vez que exige uma cooperação entre elas. A separação de tarefas entre papéis é especificada pelas regras de negócio decorrentes da política da organização.

No estudo de caso do Banco Hipotético, um empregado associado ao papel **Auditor** não poderá Agendar TED, Agendar DOC, Autorizar TED, Autorizar DOC, Efetuar pagamentos, Abrir conta corrente e nem Conceder limite a uma conta corrente, podendo apenas executar operações relacionadas com os procedimentos de auditoria das transações efetuadas. Um empregado associado ao papel **Supervisor**, por definição da política do banco, não poderá assumir o papel **Atendente**, uma vez que a política proíbe um empregado associado ao papel **Supervisor** de agendar quaisquer tipo de operações, podendo apenas autorizá-las. Estas situações ilustram relacionamentos de separação estática de deveres. Na primeira situação, em nenhuma hipótese, um empregado que assume o papel de **Auditor** poderá assumir em qualquer sessão outro papel além do papel **Funcionário**. Na segunda situação, um empregado nunca poderá assumir os papéis **Supervisor** e **Atendente**, uma vez que assim, ele poderia, por exemplo, agendar e autorizar um DOC.

A vantagem da separação estática entre papéis está na sua simplicidade, e na redução dos riscos de fraudes, erros e conflitos de interesse. Entretanto a

desvantagem de se aplicar esta regra simples está no fato de que na ausência de pessoas responsáveis pela execução de uma determinada tarefa as operações do Banco Hipotético estariam comprometidas. A tabela 5.5 apresenta os relacionamentos de separação estática (SSD) para o banco Hipotético.

Tabela 5. 5 - Relacionamentos SSD entre papéis

Nome	Papéis
SSD1	Auditor, Atendente
SSD2	Auditor, Supervisor
SSD3	Auditor, Caixa
SSD4	Supervisor, Atendente

Fonte: Elaborado pelo autor

5.5 Implementação do modelo Banco Hipotético

Inicia-se a modelagem do Banco Hipotético criando-se os papéis. A Figura 5.2 mostra a criação do papel **Atendente**. O DBA ou *Security Officer* seleciona a opção **Papéis** no menu principal, seleciona a opção **Nova** e insere o nome do papel a ser criado, na janela **Nome do Papel**.

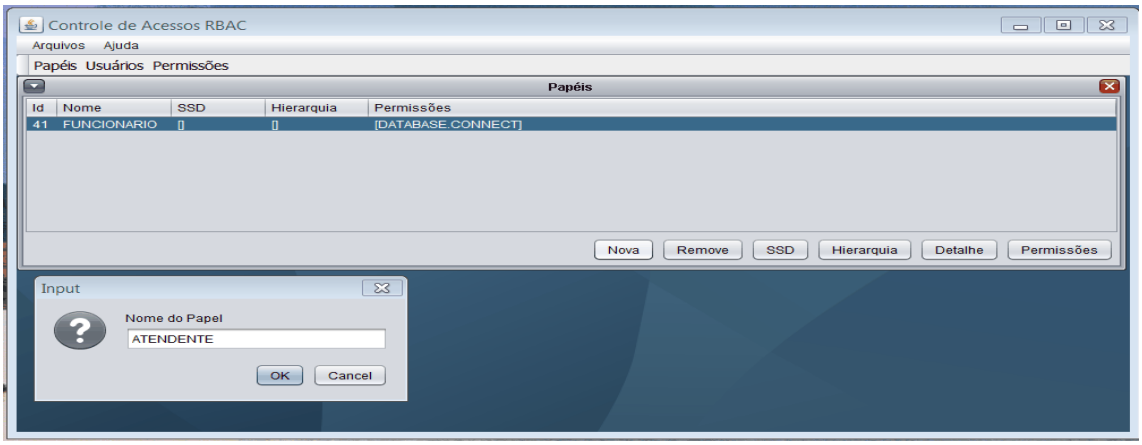

Figura 5. 2 - Criação do papel **Atendente**

Fonte: Elaborado pelo autor

O papel **Atendente** é criado no dicionário do banco de dados por meio da utilização do comando SQL *CREATE ROLE* e esta informação também é inserida na tabela da aplicação chamada **Papéis**.

A Figura 5.3 mostra a associação das permissões ao papel **Atendente.** O DBA ou *Security Officer* seleciona o papel ao qual irá associar as permissões, aciona o botão **Permissões**, e na janela **Permissões do Papel** escolhe as operações e objetos que estão associados ao papel **Atendente**. Neste caso, as permissões ***Insert*** na tabela DOC, ***Insert*** na tabela TED e ***Insert*** na tabela de conta correntes CC foram associados. As permissões são criadas no dicionário do banco de dados por meio do comando ***GRANT***. Estas informações também são gravadas, pela aplicação, na tabela **Papel_Permissões**.

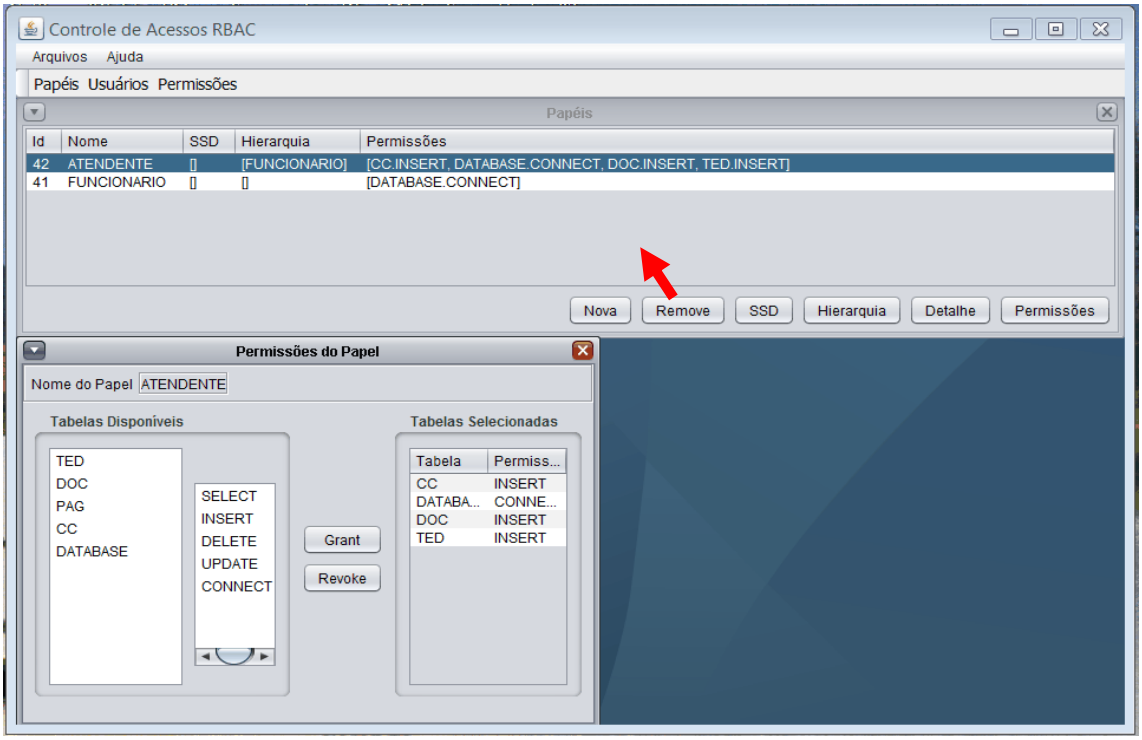

Figura 5. 3 - Associação de permissões ao papel **Atendente**
Fonte: Elaborado pelo autor

A Figura 5.4 mostra a definição da hierarquia entre os papéis **Caixa** e **Atendente**. O DBA ou *Security Officer* seleciona um papel que herda permissões de um outro papel, no caso o papel **Caixa**, aciona o botão

Hierarquia, e na janela **Hierarquia** que se abre, escolhe o papel do qual o papel **Caixa** irá herdar permissões, no caso, o papel **Atendente**.

A FCGP utilizando o algoritmo **Verifica se existe regra de separação estática entre papéis**, verifica se existe uma regra de separação estática entre **Caixa** e **Atendente**, caso exista, a ferramenta emite uma mensagem proibindo a associação. Caso não exista, a FCGP armazena esta hierarquia no dicionário do banco de dados, executando por meio da interface JDBC, um comando ***GRANT Atendente to Caixa*** diretamente no banco de dados. Em seguida, a FCGP, por meio de um comando ***INSERT***, armazena esta mesma informação na tabela do repositório responsável por armazenar hierarquias, chamada **Grupo_Papéis**.

No caso de SGBDs que possuam a opção de *trigger* de DDL, *triggers* foram criados durante a instalação da FCGP, assim o código do *trigger* associado ao comando ***GRANT*** verifica nas tabelas do repositório se existe uma separação estática entre os dois papéis e em caso afirmativo emite uma mensagem proibindo a associação, entretanto caso não exista tal restrição, o código do *trigger* insere a informação na tabela **Grupo_Papéis** e permite que o ***GRANT Atendente to Caixa*** seja executado no banco de dados.

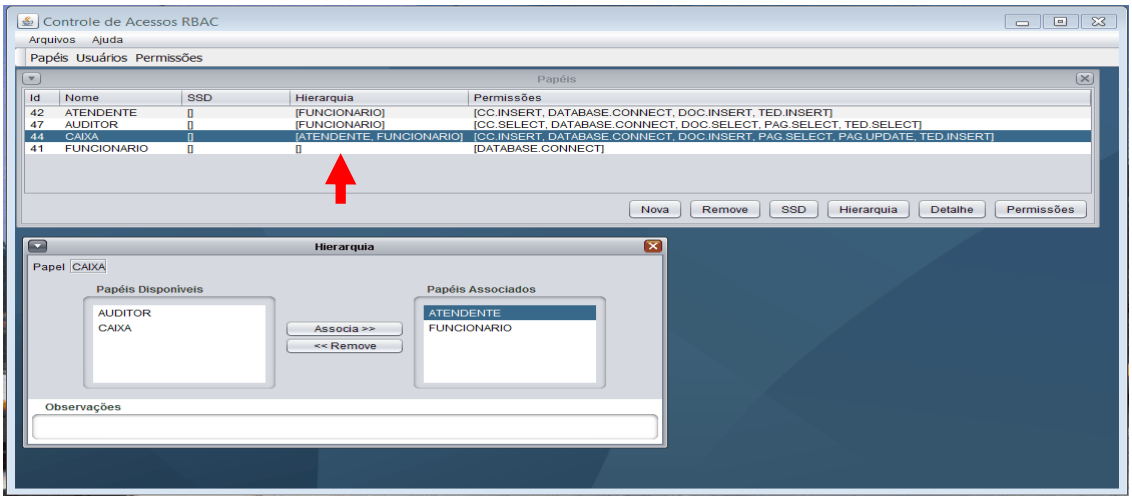

Figura 5.4: Definição de Hierarquia entre os papéis **Caixa** e **Atendente**
Fonte: Elaborado pelo autor

A Figura 5.5 mostra a definição da regra de separação estática entre os papéis **Supervisor** e **Atendente**. O DBA ou *Security Officer* seleciona o papel **Supervisor**, aciona o botão **SSD** e na tela de **Separação Estática de Papéis** escolhe os papéis que são mutuamente exclusivos ao papel **Supervisor**, no caso, o papel **Atendente**.

A FCGP utilizando o algoritmo **Verifica se existe regra de separação estática entre papéis**, verifica se já existe uma regra de separação estática entre **Supervisor** e **Atendente**, caso exista, a ferramenta emite uma mensagem proibindo a associação. Caso não exista, como o objetivo principal da FCGP é implementar a separação estática entre papéis que foi definida no padrão ANSI do modelo RBAC em SGBDs que não implementam esta funcionalidade, estas informações são armazenadas, por meio de um comando ***INSERT*** feito pela FCGP, apenas na tabela do repositório **Papel_Papéis**, e não no dicionário de dados.

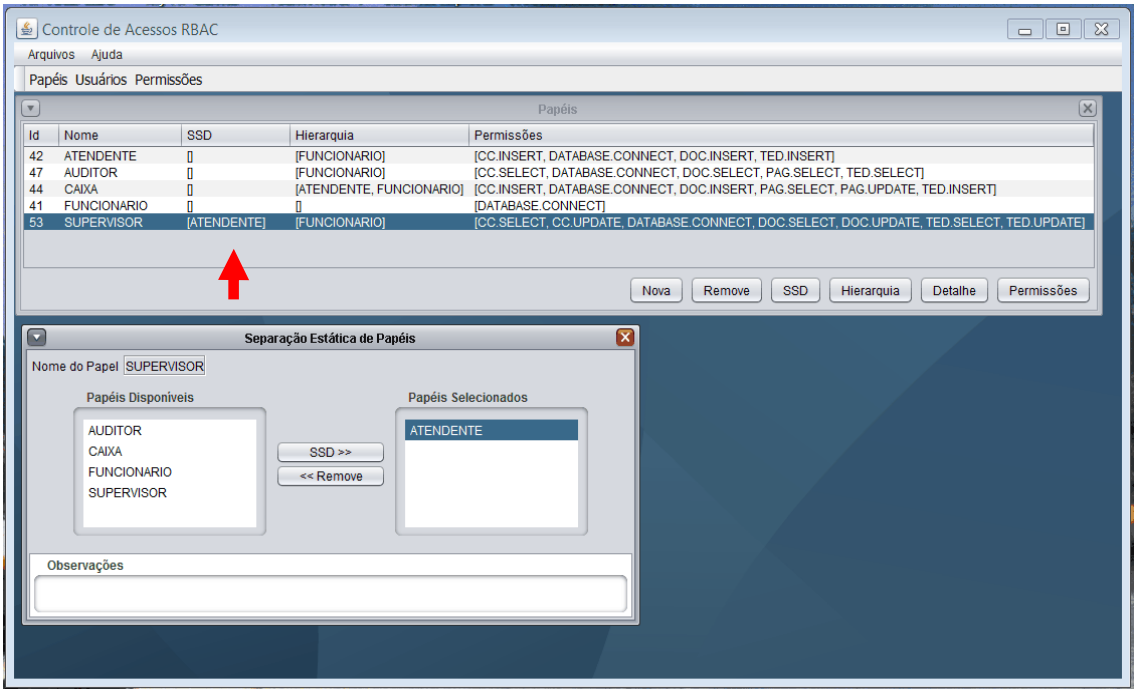

Figura 5.5: Definição de Separação Estática entre os papéis **Supervisor** e **Atendente**
Fonte: Elaborado pelo autor

Uma vez criados os papéis, as permissões, definidas as associações entre permissões e papéis, a hierarquia e as regras de separação estática entre papéis, criam-se os usuários e faz-se a associação de papéis a eles. Para o estudo de caso do Banco Hipotético, a Figura 5.6 mostra a criação do usuário **Pedro** e a associação do papel **Atendente** a ele. O DBA ou *Security Officer* seleciona a opção **Usuários** no **Menu Principal** e em seguida aciona o botão **Novo,** adicionando o nome do usuário na janela **Nome do usuário**. Em seguida, seleciona a opção **Associa** e na janela de **Papéis Associados ao Usuário** pode verificar quais os papéis já estão associados ao usuário e escolher qual outro papel deseja associar a ele, no estudo de caso, seleciona-se o papel **Atendente.**

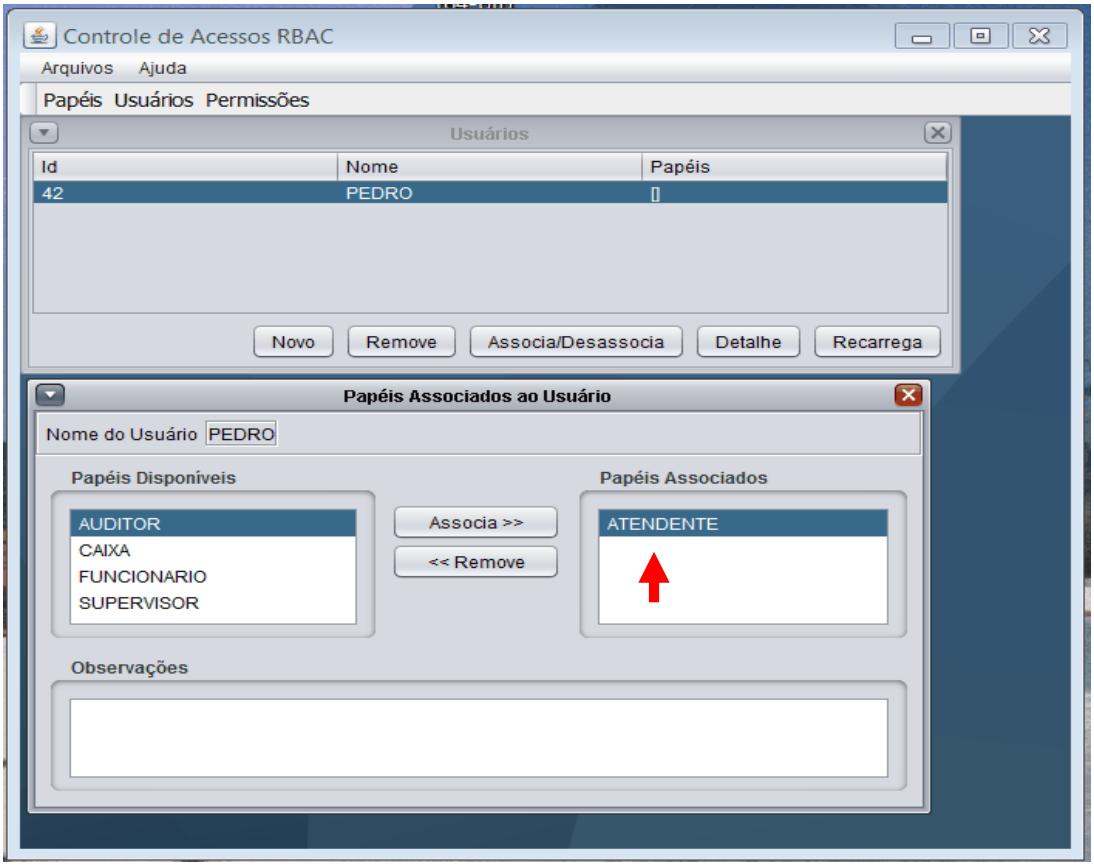

Figura 5.6: Criação do usuário **Pedro** e associação dele com o papel de **Atendente**
Fonte: Elaborado pelo autor

A FCGP utilizando o algoritmo **Verifica se existe regra de separação estática entre papéis**, verifica se existe uma regra de separação estática entre **Atendente** e algum outro papel que já esteja associado ao usuário **Pedro**, caso exista, a ferramenta emite uma mensagem proibindo a associação. Caso não exista, a FCGP armazena este usuário no dicionário do banco de dados, executando por meio da interface JDBC, um comando ***CREATE USER Pedro*** diretamente no banco de dados. Em seguida, a FCGP, por meio de um comando ***INSERT***, armazena esta mesma informação na tabela do repositório responsável por armazenar usuários, chamada **Usuários**.

No caso de SGBDs que possuam a opção de *trigger* de DDL, *triggers* foram criados durante a instalação da FCGP, assim o código do *trigger* associado ao comando ***CREATE*** verifica nas tabelas do repositório se já existe um usuário criado com este nome e em caso afirmativo emite uma mensagem proibindo a criação dele, entretanto caso este usuário ainda não exista, o código do *trigger* insere a informação na tabela **Usuários** e permite que o ***CREATE USER*** **Pedro** seja executado no banco de dados e inserido no dicionário de dados.

O DBA ou *Security Officer*, após a criação do usuário **Pedro** e da sua associação ao papel **Atendente**, recebe, por exemplo, uma solicitação para associar o papel **Supervisor** ao usuário **Pedro**, pois este recebeu uma promoção. Ao tentar fazer a associação do papel **Supervisor** ao usuário **Pedro**, o DBA ou *Security Officer* recebe uma mensagem negando tal associação devido ao fato de existir uma regra de separação estática entre estes papéis, como mostra a Figura 5.7

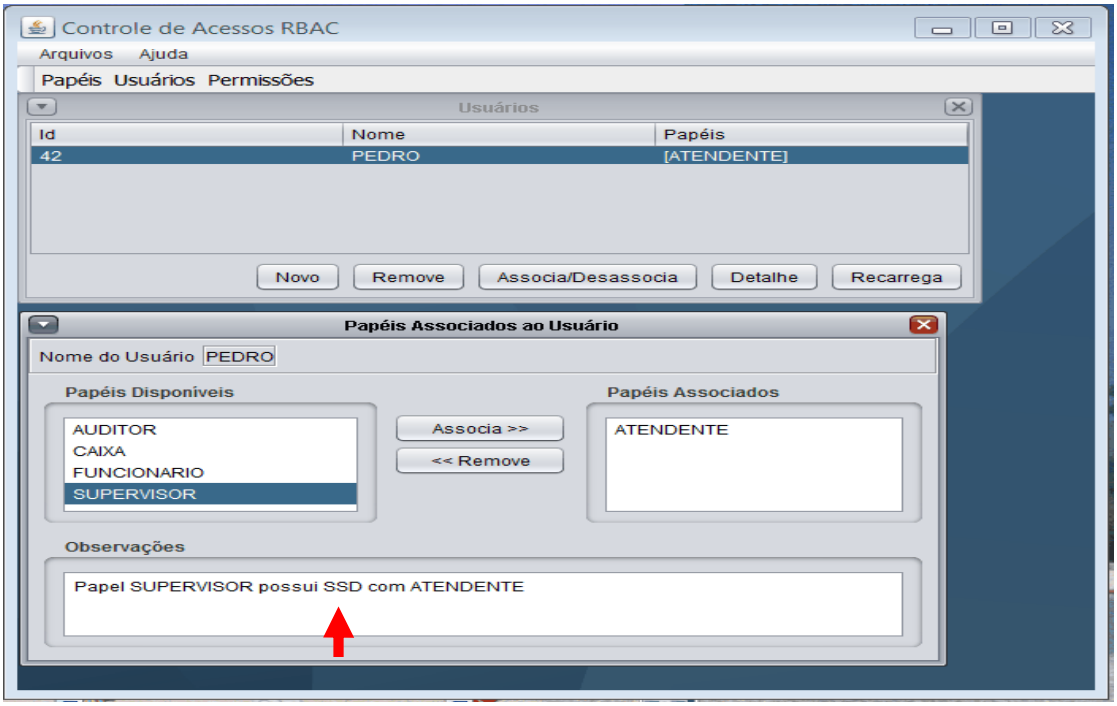

Figura 5.7: Tentativa de associação do papel **Supervisor** ao usuário **Pedro**
Fonte: Elaborado pelo autor

5.6 Considerações gerais

5.6.1 Economia do RBAC versus Discricionário

No modelo de controle de acesso discricionário as permissões são associadas diretamente aos usuários. Tendo-se um grande número de usuários e estando-se cada um deles associados a muitas permissões, tem-se um grande número de associações usuário/permissão a ser gerenciado. Assim, quando um usuário muda de cargo dentro da organização, torna-se necessária uma revisão completa das associações usuário/permissão, resultando na remoção de associações que não são mais necessárias e potencialmente perigosas e na adição de novas associações essenciais para a execução do novo cargo.

No modelo RBAC as permissões não podem ser associadas diretamente aos usuários, elas são associadas aos papéis e os papéis são associados aos usuários. Portanto, quando administrando o RBAC, dois tipos diferentes de

associações precisam ser gerenciadas, associações entre permissões e papéis e associações entre usuários e papéis. Quando um usuário muda de cargo dentro da organização, apenas as associações usuário/papel mudam. Se o cargo é representado por um único papel, então quando o cargo de um usuário muda, mudam-se apenas duas associações do tipo usuário/papel: remove-se a associação do usuário com seu papel atual e adiciona-se uma associação entre o usuário e seu novo papel.

Existe um relacionamento direto entre o custo de administração e o número de associações que precisa ser feito para se manter uma política de controle de acesso: quanto maior o número de associações, mais custoso e mais propenso a erros fica a administração. O uso do modelo RBAC reduz o número de associações que devem ser gerenciadas Ferraiolo (1999).

Um modelo econômico simples pode ser usado para se ter uma idéia aproximada da economia que se faz utilizando-se um modelo de controle de acesso baseado em papéis. Cargos ou papéis dentro das empresas normalmente são ocupados por mais de uma pessoa e a maioria deles requer mais de uma permissão a fim de que uma pessoa ocupando este cargo consiga executar as respectivas responsabilidades. Pode-se descrever as associações concedendo permissões para usuários que executam as responsabilidades de um cargo como um par ordenado consistindo de um conjunto de indivíduos e um conjunto de permissões (U,P) onde:

U = conjunto de indivíduos em um cargo;

P = conjunto de permissões necessárias para a execução de um cargo ou papel.

Assim, no modelo de controle de acesso discricionário, o número de associações necessárias para relacionar diretamente os indivíduos àquelas posições é IUI * IPI, onde

IUI = número de indivíduos no conjunto U;

IPI = número de permissões no conjunto P.

No modelo de controle de acesso RBAC, um papel pode ser descrito como um conjunto de permissões, então o conjunto P pode se referir a um papel ou a um Cargo cujas associações de usuário/papel e permissão/papel são representadas pelo par ordenado (U,P). O número de associações usuário/papel e permissão/papel necessário para autorizar cada usuário no conjunto U para cada uma das permissões no conjunto P, onde P representa um papel é IUI + IPI , ou seja, uma associação com o papel P para cada indivíduo em U e uma associação com o papel P para cada permissão em P.

Para um cargo ou papel, se IUI + IPI < IUI * IPI, então existe um vantagem administrativa do RBAC sobre a associação direta de permissões a usuários para este cargo. Uma condição suficiente para IUI + IPI < IUI * IPI é IUI, IPI > 2, que é normalmente o caso para a maioria dos cargos na maioria das organizações.

Se *n* é o número de Cargos dentro de uma organização, então a vantagem administrativa do RBAC para toda empresa, pode ser calculada com :

$$\sum_{i=1}^{n}(|Ui| + |Pi|) < \sum_{i=1}^{n}(|Ui|.|Pi|)$$

Fonte: [FER03]

Esta é apenas uma aproximação, pois usuários podem estar associados a mais de um papel e os papéis podem estar hierarquicamente relacionados.

Aplicando-se esta fórmula para o caso do banco hipotético e escolhendo-se o papel **Atendente**, tem-se que o número de indivíduos ocupando o cargo **Atendente** é igual a 2, Carlos e Ana (Tabela 5.2) , e o número de permissões necessárias para a execução do cargo **Atendente** é igual a 3 (*INSERT* na tabela **TED**, *INSERT* na tabela **DOC** e *INSERT* na tabela **CC**) (Tabela 5.3). Portanto tem-se que |U|=2 e |P|=3. Substituindo-se na fórmula tem-se que 2 + 3 < 2 * 3. Percebe-se que esta diferença seria muito maior, por exemplo, se o número de **Atendentes** fosse igual a 100, 100 + 3 < 100 * 3. Assim, comprova-se que com o uso do RBAC tem-se uma redução de custos em virtude da

redução do número de associações necessárias para manter-se a política de controle de acesso.

5.6.2 Extensões e Alternativa

Uma possibilidade de extensão deste trabalho é a implementação da separação dinâmica de deveres. Neste caso, se um usuário está incluído na lista de membros de um papel A e há um relacionamento de *DSD* entre este papel e um papel B, o usuário também pode estar incluído na lista de membros do papel B, entretanto, durante uma sessão de usuário, ele não poderá assumir simultaneamente os dois papéis. A maior dificuldade envolvida na implementação da DSD está no controle das sessões de usuários e ativações de papéis nestas sessões. Seria necessária a criação de um componente que fizesse a verificação dos papéis que se encontram em DSD a cada ativação de um papel por um usuário. Uma alternativa interessante para resolver este problema seria a criação de um componente de controle de acesso que implemente todas as funcionalidades existentes no padrão do RBAC e que fique interposto entre a aplicação e o SGBD, funcionando como o centro das decisões de controle de acesso. A Figura 5.10 mostra esta situação.

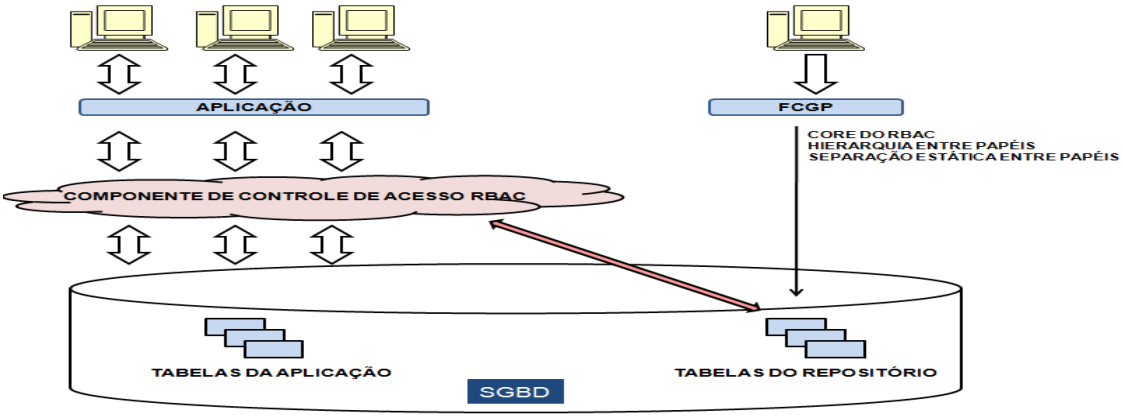

Figura 5.10: Proposta de extensão da FCGP

Fonte: Elaborado pelo autor

Nesta situação, a FCGP continua configurando e gerenciando usuários, papéis, permissões, hierarquia e restrições de relacionamento entre papéis, entretanto estas informações são armazenadas apenas nas tabelas do repositório e não no dicionário do banco de dados.

O **Componente de controle de acesso RBAC** funciona como um módulo independente e a cada momento em que é necessário controlar o acesso a alguma tabela, a aplicação pergunta ao **Componente de controle de acesso RBAC** se o usuário está autorizado a executar aquela operação sobre aquela tabela antes de realizá-la. O **Componente de controle de acesso RBAC** verifica nas tabelas do repositório gerenciado pela FCGP quais papéis possuem a permissão solicitada, e em seguida, verifica se o usuário esta associado a algum deles, ou a algum outro papel que herde as permissões do papel original. Em caso afirmativo, o **Componente de controle de acesso RBAC** responde à aplicação autorizando o acesso e inserindo o papel em uma lista de papéis ativos para aquele usuário. Neste momento, caso já exista algum papel na lista de papéis ativos do usuário que esteja diretamente ou hierarquicamente em DSD com o papel que se deseja ativar, o **Componente de controle de acesso RBAC** envia uma mensagem à aplicação negando o acesso.

A principal diferença deste modelo com relação à FCGP é que as aplicações não acessam diretamente o banco de dados e nem os papéis são configurados dentro dele. O **Componente de controle de acesso RBAC** realizará toda a verificação de usuários, papéis, permissões, hierarquia e restrições de relacionamentos entre papéis (estática e dinâmica) com base na informação contida no repositório. Dentre as principais vantagens neste caso tem-se que outros tipos de restrições de relacionamento entre papéis poderão ser criados, como por exemplo, restrições temporais, em que um papel possui um determinado tempo de validade; cardinalidade de papéis, em que um papel pode ser associado a um número determinado de usuários, e ainda não será mais necessário manter-se a consistência entre as tabelas do repositório e o dicionário de dados. Uma das desvantagens desta alternativa é que alguma

forma de controle de acesso discricionário deve ser implementada no banco de dados uma vez que os papéis não serão criados dentro dele. Com isso, duas verificações de controle de acesso serão feitas, uma pelo **Componente de controle de acesso RBAC** e outra pelo banco de dados.

5.7 Conclusões

Este capítulo apresentou como a FCGP pode ser utilizada para implementar o modelo de controle de acesso baseado no RBAC de uma instituição financeira hipotética e como ela permite a criação de uma regra de separação estática de deveres entre os papéis **Supervisor** e **Atendente**, impedindo assim, associações indevidas fraudulentas ou acidentais.

Um modelo econômico simples foi utilizado para o caso da instituição financeira hipotética para se ter uma idéia aproximada da economia que se faz utilizando-se um modelo de controle de acesso baseado em papéis em relação ao modelo discricionário. Quanto maior o número de indivíduos exercendo um mesmo cargo, e quanto maior o número de permissões necessárias para se exercer este cargo, maior a vantagem do RBAC com relação ao DAC.

O capítulo apresentou também uma sugestão de trabalho futuro no qual aplicações de usuários acessam um **Componente de controle de acesso RBAC** ao invés do banco de dados diretamente. O **Componente de controle de acesso** utilizando as informações de usuários, permissões, papéis, hierarquia e restrições entre papéis fornecidas pela FCGP faz todo o controle de acesso, incluindo a separação dinâmica entre papéis a cada solicitação de operação sobre tabelas deste banco de dados feita pela aplicação.

6 CONCLUSÕES

6.1 Resumo

Este trabalho apresenta a FCGP, o protótipo de uma ferramenta para configuração e gerência de papéis baseada no modelo RBAC. O objetivo principal da FCGP é implementar a separação estática de papéis, que evita que um mesmo usuário seja associado a papéis que tenham sido definidos como mutuamente exclusivos por apresentarem algum risco ao negócio. A FCGP implementa o *core* do RBAC, a hierarquia e a separação estática entre papéis utilizando um conjunto de algoritmos, interfaces e tabelas próprias, e propaga estas informações para o dicionário do banco de dados por meio de comandos SQL. O controle de acesso das aplicações ao banco de dados continua sendo feito pelo banco de dados, entretanto como a gerência dos usuários, papéis, permissões, hierarquia e separação estática entre papéis são feitas pela FCGP, ela evita que um usuário seja associado a papéis mutuamente exclusivos com base na informação contida em suas tabelas, evitando assim que uma associação deste tipo seja propagada para o banco de dados.

Para manter a consistência entre as informações do dicionário de dados e as tabelas do repositório e para evitar que papéis mutuamente exclusivos sejam associados a usuários por meio de outras ferramentas, a FCGP cria *triggers* de DDL, em SGBDs que suportem esta funcionalidade.

O resultado dos testes realizados com a implementação do estudo de caso do Banco Hipotético mostrou que a ferramenta impede a associação de papéis definidos como mutuamente exclusivos a um mesmo usuário, evitando assim associações acidentais ou fraudulentas.

6.2 Análise Geral e contribuições

Entre os modelos de controle de acesso mais conhecidos, DAC, MAC e RBAC, este último tem-se tornado o mais utilizado, passando a ser implementado em uma grande variedade de produtos, como sistemas

operacionais, sistemas gerenciadores de bancos de dados, redes de computadores e produtos de administração de segurança. Entretanto, embora os sistemas gerenciadores de bancos de dados tenham sido os primeiros produtos comerciais a suportarem o RBAC, nem todas as funcionalidades definidas no seu padrão ANSI são implementadas por eles. As funcionalidades mais comumente implementadas são o *core* do RBAC e a hierarquia entre papéis.

O trabalho contribui para mostrar a importância da existência de regras de separação entre papéis dentro de uma organização, com o objetivo de evitar ações fraudulentas e por permitir a implementação destas regras em SGBDs que não possuam esta funcionalidade.

Outra contribuição importante foi mostrar que a administração de uma política de controle de acesso baseada no modelo RBAC, como a realizada pela FCGP, é muito menos custosa e propensa a erros do que uma baseada no modelo discricionário, pois com o RBAC, as permissões não são associadas aos usuários individualmente. Ao invés disso, permissões são associadas aos papéis, um conjunto bem menor que o de usuários em grandes organizações. Novas associações entre permissões e papéis podem ser criadas enquanto antigas são removidas na medida que as funções organizacionais mudam e evoluem. Esse relacionamento entre os papéis do modelo RBAC e os papéis exercidos na organização facilita o gerenciamento de permissões: o DBA ou *Security Officer* pode atualizar os papéis sem ter que atualizar as permissões para cada usuário do sistema individualmente, como no modelo discricionário.

6.3 Sugestão para pesquisas futuras

Por fim, foi deixado a cargo de um trabalho futuro o desenvolvimento de um módulo de controle de acesso que implemente todas as funcionalidades definidas no padrão do RBAC usando as informações armazenadas nas tabelas do repositório da FCGP e que possa ser integrado as aplicações. Neste caso, a cada momento em que é necessário controlar o acesso a algum recurso, as

aplicações solicitam ao módulo de controle de acesso e não ao banco de dados, se o usuário está autorizado a executar aquela operação ou não, antes de realizá-la. Dessa forma, o controle de acesso passa a ser feito por este módulo e não pelo banco de dados. A grande vantagem de um módulo separado é a possibilidade de implementação não apenas das funcionalidades existentes no padrão, mas também de outras funcionalidades, como a cardinalidade de papéis e as restrições temporais de acesso.

REFERÊNCIAS

ASOKAN N. et al. The State of the Art in Electronic Payment Systems. **IEEE Computer**, p. 28-35, Sep.1997.

BELL, D. E.; LAPADULA L.J. **Secure Computer Systems: Mathematical Foundations and Model**. Bedford, MA: The Mitre Corporation, 1973.

BREWER, D.; NASH, M.The Chinese wall security policy**.** In: **IEEE SYMPOSIUM ON SECURITY AND PRIVACY,** 1989, Oakland. **Proceedings…** Oakland : IEEE Press, 1989. P.206.

CHANDRAMOULI, R.; SANDHU, R. **Role Based Access Control Features in Commercial Database Management Systems.** 21st National Information Systems Security Conference, October 6-9, 1998, Crystal City, Virginia.

CRAMPTON, J.; LOIZOU, G. Administrative scope: A foundation for role-based administrative models. **ACM Transactions on Information and System Security (TISSEC)**, New York, v.6, p.201-231, Nov. 2003.

DE CAPITANI DI VIMERCATI, S. et al. Access Control: Principles and Solutions. **Software-Practice & Experience,** New York, v.33, p.397-421, 2003.

DIRX IDENTITY 7.0. 2004. Disponível em: <http://solutions.fujitsu-siemens.com/sotware,catalog/product.php?id=20000628&lang=en&platform=all> . Acesso em: 09 mar. 2007.

ELMASRI, R.; NAVATHE, S.B. **Fundamentals of Database Systems**. 4.ed. Addison-Wesley, 2003.

FERRAIOLO D.; BARKLEY J.; KUHN D.R. A role-based access control model and reference implementation within a corporate intranet. **ACM Transactions on Information and System Security**, v.2, p.34-64, Feb.1999.

FERRAIOLO, D. et al. Proposed NIST Standard for Role Based Access Control. **ACM Transactions on Information and System Security**, v. 4, n.3, p. 224-274, Aug. 2001.

FERRAIOLO, D., KUHN, D.R. Role-Based Access Control. In:15TH NIST - NCSC NATIONAL COMPUTER SECURITY CONFERENCE, Oct. 13-16, 1992, Baltimore. **Proceedings…** Baltimore: NIST.1992. p. 554-563. Disponível em: <http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-92.pdf>. Acesso em: 24 mar. 2009.

FERRAIOLO, D.; CUGINI, J.A; KUHN, D.R. Role-Based Access Control (RBAC):Features and Motivations. In*:* 11th Annual Computer Security Applications. 11., 1995*,* New Orleans*.* **Proceedings...** New Orleans: NIST.1995.

FERRAIOLO, D.; GILBERT D.; LYNCH N. An Examination of Federal and Commercial Access Control Policy Needs. In: NIST-NSA NATIONAL (USA) COMPUTER SECURITY CONFERENCE, 16, 1993, Baltimore. **Proceedings…** Maryland: NIST, 1993. p.107-116.

FERRAIOLO, D.; KUHN, D.R.; CHANDRAMOULI, R. **Role-Based Access Control**. 2.ed. Norwood: Artech House, 2007.

GALLAHER, MICHAEL P. et al. **NIST Planning Report 02-1 - The economic impact of role-based access control**. National Institute of Standards & Technology. Program Office strategic planning and economic analysis group. NIST. March. 2002.

LAMPSON, B. W., **Dynamic Protection Structures**, **Proceedings of the November 18-20 .** New York : AFIPS : American Federation of Information Processing Societies, v.35, p.27-38, Nov. 1969.

SAMARATI P.; DE CAPITANI DI VIMERCATI S. **Access control: Policies, Models and mechanisms.** International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design, LNCS 2171. Springer-Verlag : London, 2001. p. 137-196. Disponível em: <http://spdp.dti.unimi.it/papers/sam-fosad.pdf>. Acesso em : 09 mar. 2007.

SANDHU R.S.; SAMARATI P. S. Access Control: Principles and Practice. **IEEE Communications**, v. 32, p. 40-48, Sep.1994.

SANDHU, R.S. et al. Proposed NIST standard for role-based access control. **ACM Transactions on Information and System Security ,** New York, v.4, n.3, p.224-274, Aug. 2001.

SANDHU, R.S.; COYNE,E.J. et al. Role-Based Access Control Models. **IEEE Computer**, Washington, v.29, n.2, p.38-47, Feb.1996.

SANDHU, R.S.; FERRAIOLO, D.; AND KHUN D.R. The NIST Model for Role-Based Acess Control: Towards a Unified Standard. In: 5th ACM Workshop on Role Based Access Control, 5, Berlin, 2000, Berlin. **Proceedings...** Berlin: ACM, 2000**.** p.47-63.

REFERÊNCIAS CONSULTADAS

HOFFMAN, J. Implementing RBAC on a Type-Enforced System. In: 13th Annual Computer Security Applications Conference (ACSAC 1997), 8-12, 1997, San Diego. **Proceedings...** San Diego, 1997. p.158

TIDSWELL.J.; J.POTTER. An Approach to Dynamic Domain and Type Enforcement. In: Second Australasian Conference on Information Security and Privacy, 2, 2000, Berlin. **Proceedings…** Berlin: Springer, 2000. p. 26-37