Pós-Graduação em Ciência da Computação

UMA METODOLOGIA PARA CONTROLE DE ACESSO GRANULAR EM SISTEMAS DE BANCO DE DADOS RELACIONAIS *OPEN SOURCE*

Por

MAURÍCIO GUEDES MARQUES

Dissertação de Mestrado Profissional

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, ABRIL/2013

MAURÍCIO GUEDES MARQUES

UMA METODOLOGIA PARA CONTROLE DE ACESSO GRANULAR EM SISTEMAS DE BANCO DE DADOS RELACIONAIS *OPEN SOURCE*

Orientador: Prof. Dr. Fernando da Fonseca de Souza

Dissertação apresentada à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

RECIFE, ABRIL DE 2013.

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Marques, Maurício Guedes
Uma metodologia para controle de acesso granular em sistemas de banco de dados relacionais open source. / Maurício Guedes Marques. - Recife: O Autor, 2013.
viii, 120 folhas: fig., quadro

Orientador: Fernando da Fonseca de Souza.
Dissertação (mestrado profissional) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia e anexo.

1. Banco de dados. 2. Segurança de dados. I. Souza, Fernando da Fonseca de (orientador). II. Título.

025.04 CDD (23. ed.) MEI2013 – 062

Dissertação de Mestrado Profissional apresentada por **Mauricio Guedes Marques** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título**, "Uma metodologia para controle de acesso granular em sistemas de banco de dados relacionais open source",** orientada pelo **Professor Fernando da Fonseca de Souza** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Ana Carolina Brandão Salgado
Centro de Informática / UFPE

_____

Prof. Carlos Eduardo Santos Pires
Universidade Federal de Campina Grande

_____

Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 18 de março de 2013.

_____

Profª. EDNA NATIVIDADE DA SILVA BARROS
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

DEDICATÓRIAS

Dedico este trabalho a todas as pessoas que, de alguma forma, contribuíram para que ele fosse concluído. Em especial a minha amada esposa Florrie Fernandes, minha linda filha Maria Clara e aos meus pais e ídolos Luiz Maurício e Norma Guedes.

AGRADECIMENTOS

Agradeço a Deus por todas as bênçãos recebidas nos últimos anos.
Agradeço a minha família, minha esposa Florrie por existir, sempre me apoiar em tudo o que faço e superar todos os sábados de aula. Obrigado também ao meu irmão Artur, minha sogra Avani, minha amada filha Maria Clara, meu pai Luiz Maurício e minha mãe Norma, obrigado pela dedicação e atenção dispensado por mim a vida inteira. Hoje, sendo pai, compreendo, reconheço e agradeço tudo o que vocês sempre fizeram por mim.
Agradeço aos meus afilhados Lucas Lins e Luan Lins por cobrarem um espaço dedicado a eles neste documento. O padrinho não se esqueceu de vocês.
Agradeço a meu orientador Fernando Fonseca. Sou grato pelos ensinamentos, paciência e apoio.
Agradeço aos meus amigos do mestrado, em especial a Bruno Pimentel, Luiz Fernando e Magno Gouveia. Obrigado pessoal pelo apoio e pelo tempo que passamos juntos.
Agradeço a meu sócio, amigo e parceiro Alexandre Anselmo por segurar a barra de trabalho no período do mestrado.
Ao meu "consultor Java" Gustav Monteiro por toda força e ajuda no protótipo.
Aos amigos da Linx Sistemas. Obrigado pessoal.
Enfim muito obrigado a todos. Vocês me ajudaram e chegar ao fim desta jornada.
Muito obrigado.

.

UMA METODOLOGIA PARA CONTROLE DE ACESSO GRANULAR EM SISTEMAS DE BANCO DE DADOS RELACIONAIS *OPEN SOURCE*

RESUMO

O controle e manutenção do acesso granular aos dados, ou seja, no nível de linha e coluna, sobretudo em organizações que dispõem de diversos sistemas de banco de dados distintos, é um problema em aberto, em que há poucos estudos de como resolver o problema a partir de um método ou solução única para o controle, de forma corporativa, da segurança no acesso aos dados das diversas bases disponíveis.

As soluções presentes atualmente no mercado estão disponíveis apenas para sistemas de gerenciamento de banco de dados (SGBD) cujo licenciamento muitas vezes impede a utilização deste recurso. Além disso, tais soluções são proprietárias e só funcionam em um único SGBD.

Muitas aplicações utilizam o conceito de visões (*views*) para prover o controle de acesso granular. No entanto, apesar de flexível, não é uma solução prática porque as lógicas das visões podem se tornar bastante complexas para garantir o controle de acesso, dificultando o gerenciamento e manutenção das mesmas. Além disso, é uma solução com baixa escalabilidade quando existe um número grande de usuários e perfis de acesso. Desta forma, grande parte das aplicações implementa o controle de acesso dentro do código da aplicação. Isso proporciona uma série de desvantagens e de falhas de segurança associadas.

Este trabalho traz uma proposta de metodologia para controle de acesso granular em sistemas de banco de dados relacionais *open source*. Esta solução combina sistemas gerenciadores de banco de dados relacionais *open source* com a intervenção dos comandos SQL através do *driver* de acesso ao banco de dados. O *driver* customizado lê e aplica, implicitamente, políticas de segurança previamente armazenadas em um banco de dados embarcado, alterando o comando SQL e, consequentemente, o resultado apresentado ao usuário final. Um protótipo foi desenvolvido como prova de conceito e os experimentos foram baseados no SGBD MySQL usando o *driver* JDBC. Os resultados obtidos foram analisados como positivos visto que não houve perda de desempenho significativa na interceptação e reescrita do Comando SQL nos experimentos realizados.

Para tanto, este trabalho aborda os principais conceitos de segurança da informação, segurança de banco de dados, modelos de autorização e os principais mecanismos de controle de acesso utilizados em sistemas de banco de dados relacionais como o DAC, MAC e RBAC. Além disso, foram abordados o conceito de controle de acesso granular aos dados bem como os principais mecanismos utilizados no mercado. Após isto, são analisados alguns estudos que apresentam alternativas para o problema do controle de acesso granular aos dados. Por fim, são apresentadas as conclusões e sugestões de trabalhos futuros dentro da área de estudo.

Palavras-chave: Banco de dados, Segurança de dados, Políticas de Segurança, Controle de Acesso.

A METHODOLOGY TO FINE GRAINED ACCESS CONTROL IN OPEN SOURCE RELATIONAL DATABASE SYSTEMS

ABSTRACT

The control and maintenance of granular access to data at the level of column and row, especially in organizations that make use of several distinct database systems, is an open problem with few studies on how to solve the problem using only a method or unique solution for the control, in a corporative way, of security in the access to the data of the diverse available bases. Solutions currently in the market are available only for database management systems (DBMS) whose licensing many times impairs the use of this resource. Moreover, such solutions are proprietary and just work for a given DBMS.

Many applications use the concept of views to provide the granular control access. However, although flexible, it is not a practical solution because the logic of the views can become complex in order to guarantee the access control, making it difficult to manage and to maintain such mechanisms. Moreover, it is a solution with low scalability when it considers a large number of users and access profiles. In such a way, most of the applications implement the access control inside the code of the application. This may causes a series of disadvantages and failures to security.

This work proposes a methodology for granular access control to open source relational database systems. This solution combines open source relational database management systems with the intervention of SQL commands through the access driver to the database. The customized driver reads and applies (implicitly) security policies previously stored in a built in data base, modifying SQL command accordingly and so the final result presented to the end user. A prototype was developed as proof of concept and experiments were performed on the MySQL DBMS, using the JDBC driver. The results were analyzed as positive because there was no significant loss of performance in intercepting and rewriting the SQL command in the experiments.

To achieve that, this work focused on the main concepts of information security, database security, models of authorization and the main mechanisms used for access control to relational database systems such as DAC, MAC and RBAC. Moreover, the concept of granular access control to data as well as the main mechanisms used in the market was focused. Following this, it was analyzed some studies that present alternatives to solve the problem of granular access control to data. Finally, conclusions and suggestions for future works in this area were presented.

Keywords: Database, Data Security, Security Policy, Fine Grained Access Control.

Sumário

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

PRINCIPAIS ABREVIATURAS E SIGLAS

ANSI	American National Standards Institute
API	Application Program Interface
C++	Linguagem de programação multi-paradigma e de uso geral
CIA	Central Intelligence Agency
CPOL	Code Project Open License
DAC	Discretionary Access Control
DML	Data Manipulation Language
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
MAC	Mandatory Access Control
MER	Modelo Entidade-Relacionamento
ODBC	Open Data Base Connectivity
P3P	Platform for Privacy Preferences
RACF	Resource Access Control Facility
RBAC	Role Based Access Control
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
UML	Unified Modeling Language
VPD	Virtual Private Database
WEB	World Wide Web

Capítulo 1: Introdução

Como as organizações aumentaram significativamente a adoção de sistemas de banco de dados como principal tecnologia para gerir e armazenar dados no dia-a-dia das principais operações e de tomada de decisão, a segurança dos dados geridos por estes sistemas torna-se crucial. O uso indevido ou perda de dados afetam não apenas um único usuário ou aplicação, mas pode ter consequências desastrosas para toda a organização (Bertino e Sandhu, 2005).

A aplicação de políticas de segurança para sistemas computacionais em mecanismos de controle de acesso é um campo vasto e variado. O objetivo fundamental de qualquer mecanismo de controle é o de proporcionar um sistema verificável para garantir a proteção de informações não autorizadas e acesso inadequado conforme descrito em uma ou mais políticas de segurança. Em geral, tais políticas devem prover confidencialidade (capacidade de proteger a divulgação de informações) ou integridade (capacidade de controlar a modificação de informações) (Ausanka-Crues, 2006).

Neste contexto, o controle de acesso granular aos dados, que permite limitar o acesso a dados no nível de linha e colunas específicas, é uma necessidade prática em todas as aplicações que utilizam sistemas de banco de dados (Kabra et al, 2006).

A primeira proposta de implementação para o controle de acesso granular aos dados foi realizada por Stonebraker e Wong (1974). O objetivo inicial era incorporar o mecanismo de controle de acesso granular aos dados baseado na modificação dinâmica das consultas para o sistema de banco de dados INGRES (Actian Corporation, 2012). Desde então, várias metodologias foram propostas (Agrawal et al, 2005; Surajit et al, 2007; Opyrchal et al, 2011) e vários mecanismos foram criados de maneira proprietária pelos principais fornecedores de banco de dados (Oracle Database Security Guide, 2011; Sybase Technical Library, 2012). No entanto, a existência de estudos recentes, limitações dos métodos propostos e o alto valor de licenciamento das soluções de mercado indicam que o problema relacionado ao controle de acesso granular aos dados ainda está em aberto.

Esta dissertação aborda a pesquisa realizada sobre o controle de acesso granular aos dados, aplicados a banco de dados relacionais, com foco em uma nova abordagem baseada na interceptação e reescrita dos comandos SQL realizada não diretamente pelo

SGBD, e sim, pelo mecanismo de conexão ao banco de dados. A nova abordagem é utilizada para prover o controle de acesso granular aos dados voltado para sistemas gerenciadores de banco de dados *open source*. Desta forma, além de centralizar as políticas de acesso de vários sistemas de informação em um único "repositório", é possível diminuir os custos de licenciamento de software referentes às soluções já disponibilizadas no mercado.

1.1 Motivação

A segurança das informações de uma organização está diretamente ligada ao controle de acesso que os funcionários possuem nos vários sistemas de informação disponíveis na empresa. Os sistemas de banco de dados vêm sendo, nas últimas décadas, largamente utilizados para armazenar dados de uma grande quantidade de aplicações de computador e têm sido escolhidos como forma de armazenamento dos sistemas de informação e aplicações onde o controle e a coordenação entre muitos usuários são necessários.

Os principais fornecedores de software ainda possuem o desafio de garantir a segurança, no nível granular, das informações contidas no banco de dados. O controle de acesso granular consiste em prover diferentes tipos de acesso (leitura, escrita ou execução) para um usuário ou grupo de usuários. Isto pode acontecer também na forma de predicados através da modificação do comando SQL em tempo de execução (Bertino et al., 2005; Wang et al., 2007; Shi e Zhu, 2010).

Particularmente, algumas empresas trabalham com vários sistemas de banco de dados, sendo necessário um controle de acesso único, ou seja, que não utilize tecnologia, características e, principalmente, que não sejam obrigados a ter alto custo de licenciamento das soluções atuais oferecidas pelos fornecedores dos Sistemas de Gerenciamento de Banco de Dados (SGBD) como, por exemplo, o VPD (Virtual Private Database) da Oracle (Oracle Database Security Guide, 2012) ou o Multi Level Security do DB2 (Bird, 2000). Além disso, sistemas de banco de dados como o MySQL (MySQL, 2012) e o H2 (H2 Database Engine, 2012) não dispõem de características nativas do SGBD para tal controle. Torna-se necessária, portanto, desenvolver uma forma de garantir a segurança no acesso granular aos dados de qualquer sistema de informação que utilize bancos de dados relacionais para persistência dos dados.

1.2. Objetivos

O objetivo principal deste trabalho é propor uma metodologia de controle de acesso granular aos dados de SGBD relacional *open source* e que atenda aos requisitos de segurança aos dados e informações das organizações. Aliado a isto, será criada uma ferramenta que realize a geração desse modelo, e que sirva como prova de conceito para o método proposto. Tal ferramenta será experimentada no SGBD MySQL (MySQL, 2012). Para tanto, é necessário mostrar a importância da segurança de dados, relacionando-a aos principais tipos de controle de acesso no contexto de banco de dados. Como objetivos específicos, tem-se:

I. Definir a arquitetura necessária para elaboração de uma metodologia que contemple os requisitos necessários para o controle de acesso granular a dados;
II. Propor uma metodologia para controle de acesso granular a dados de SGBD relacionais *open source*; e
III. Implementar, utilizando recursos de software livre, um protótipo que utilize a metodologia proposta para controle de acesso granular a dados de SGBD relacionais *open source*.

.

1.3. Estrutura da dissertação

Esta dissertação, além deste capítulo introdutório, é composta pelos sete capítulos descritos a seguir.

No Capítulo 2, é mostrada uma visão geral dos conceitos básicos sobre segurança de banco de dados tais como segurança da informação, segurança em banco de dados, modelos de autorização e o capítulo será finalizado com a descrição dos principais mecanismos de controle de acesso.

No Capítulo 3, será abordado o conceito de controle de acesso granular. Nele, as principais utilizações bem como algumas das tecnologias atualmente existentes nos principais SGBD disponíveis no mercado são consideradas.

No Capítulo 4, é apresentada uma explanação acerca de algumas metodologias existentes para realizar o controle de acesso granular aos dados. São mostrados, também,

até que ponto elas evoluíram e quais lacunas em aberto se pretende explorar. A partir daí, é apresentada a definição do modelo proposto para controle de acesso granular aos dados.

No Capítulo 5, é descrita a metodologia proposta por esta dissertação. Nele, são apresentadas a definição formal e base conceitual da referida metodologia utilizada para tornar o acesso aos dados seguros. Além disso, são também apresentados os algoritmos criados para alcançar a solução proposta.

No Capítulo 6, é apresentada a ferramenta "Grão" como prova de conceito para a metodologia descrita no capítulo anterior. Sua arquitetura, diagrama de componentes, modelo de dados e principais funcionalidades são detalhadas. Além disso, são descritas as tecnologias utilizadas e é mostrado um exemplo de utilização do fluxo da ferramenta para controle de acesso granular aos dados. Somado a isto, são detalhados os experimentos realizados e resultados obtidos utilizando a ferramenta criada para comprovar sua eficácia e eficiência.

O Capítulo 7 destaca as principais contribuições e limitações desse trabalho. Nele, também é realizada uma análise dos resultados alcançados, assim como são feitas propostas e recomendações para trabalhos futuros, contemplando sugestões e tópicos que não foram abordados nesta dissertação.

Finalmente, as referências bibliográficas utilizadas são listadas.

A dissertação conta ainda com quatro apêndices: o Apêndice A apresenta um exemplo de utilização da solução para o controle de acesso granular aos dados disponibilizado pela Oracle chamado Virtual Private Database (VPD). O Apêndice B apresenta um exemplo de utilização da solução para o controle de acesso granular aos dados disponibilizado pelo Sybase chamado Row-Level Authorization. O Apêndice C lista um exemplo de utilização da solução para o controle de acesso granular aos dados disponibilizado pelo DB2 chamado Multiple Level Security. Finalmente o Apêndice D detalha o diagrama entidade relacionamento utilizado por alguns módulos do protótipo criado.

Capítulo 2: Conceitos Básicos

Neste capítulo é feita uma revisão dos principais conceitos relativos à segurança de informação e de dados, bem como tópicos relevantes como os principais requerimentos de segurança, modelos de autorização e tipos de controles de acesso. Todos estes assuntos são fundamentais como referenciais conceituais para o entendimento dos próximos capítulos e para a construção de uma metodologia de controle de acesso granular aos dados, que é objetivo desta dissertação.

2.1. Segurança da Informação

A palavra segurança tem origem no latim e significa "sem preocupações", ou seja, segurança é ausência de risco, a previsibilidade, a certeza quanto ao futuro.

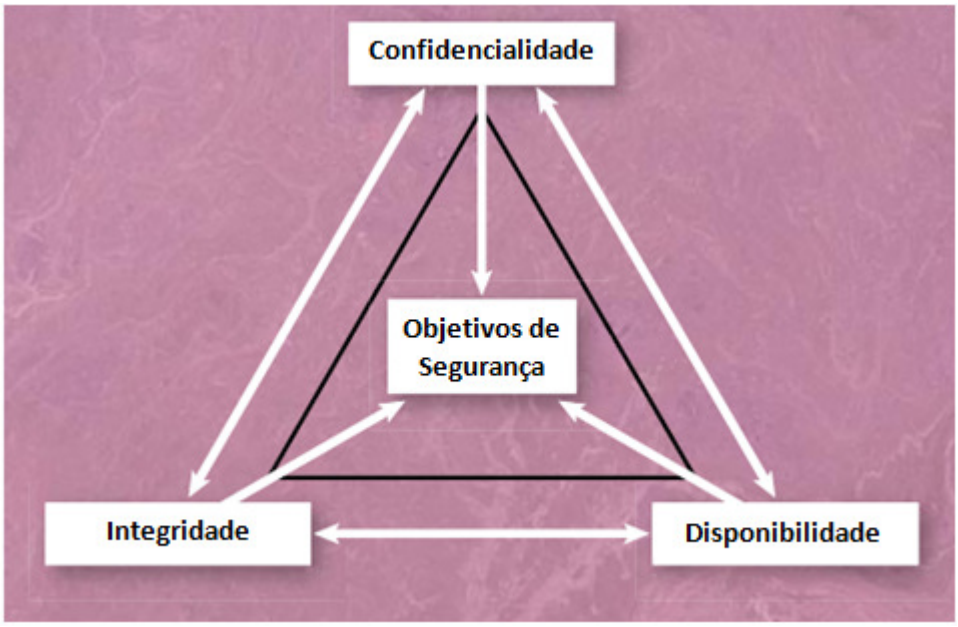
No que diz respeito à segurança da informação, a norma ISO/IEC 27002 (ABNT NBR ISO/IEC 27002, 2005) trata de aspectos específicos de segurança da informação e foi publicada pela ISO - *International Organization for Standardization* (ISO, 2012) e pelo IEC - *International Electrotechnical Commission* (IEC, 2012). Com o título "Tecnologia da Informação - Técnicas de segurança - Código de prática para a gestão da segurança", a ISO/IEC 27002 conceitua:

- **Informação** - é um ativo que, como qualquer outro ativo importante para negócios, tem um valor para a organização e, consequentemente, necessita ser adequadamente protegida; e
- **Segurança da informação** - A proteção da informação de vários tipos de ameaças para garantir a continuidade do negócio, minimizar os riscos ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio.

Ainda de acordo com a norma citada, a segurança da informação é obtida a partir da execução de um conjunto de controles adequados, incluindo políticas, processos, procedimentos, estruturas organizacionais e funções de hardware e software. Face ao exposto, pode-se verificar que o conceito de segurança da informação é bastante amplo. Desta forma, este trabalho foca especificamente a subárea de segurança da informação que trata da segurança de dados e controle de acesso aos referidos dados. Os requisitos de segurança de dados são fundamentais para garanti-la. Segundo os padrões

internacionais da norma ISO/IEC 27002, tais requerimentos podem ser agrupados em três categorias principais, também conhecida como tríade da CIA conforme a Figura 2.1:

Figura 2.1. Categorias de Segurança conhecidas como tríade da CIA



Fonte: Stone e Merrion, 2004.

O conjunto de requerimentos ou propriedades visualizado na Figura 2.1 é utilizado na literatura em várias definições para segurança (Gollman, 1999; Harris, 2002). De acordo com a norma ISO/IEC 27002, cada propriedade tem a função conforme descrito abaixo:

- **Confidencialidade** – Necessidade de proteger informações consideradas sensíveis de forma que os dados não sejam vistos indiscriminadamente. Isto significa permitir que usuários só vejam o que devem realmente ver;
- **Integridade** – Propriedade que tem como objetivo assegurar que os recursos ou dados estejam protegidos contra exclusão e corrupção tanto no banco de dados quanto no tráfego pela rede; e
- **Disponibilidade** – Tem como objetivo prover a disponibilidade do sistema para os usuários. Aspectos como resistência a falhas, escalabilidade e flexibilidade devem ser respeitados.

2.2. Segurança de Banco de Dados

A segurança de banco de dados compreende uma série de medidas, políticas e mecanismos para garantir a confidencialidade, integridade e disponibilidade dos dados e proteger o sistema de possíveis ataques que podem ser realizados por pessoas autorizadas ou não, de maneira intencional ou acidental (Dastjerdi et al, 1996).

Cada SGBD fornece uma série de funcionalidades para garantir a segurança do banco de dados. No entanto, existe uma série de políticas ou requerimentos que todo sistema de banco de dados deve fornecer, no que diz respeito ao aspecto da segurança. Tais características ou políticas de segurança foram descritas inicialmente nos estudos de Castano (1995) e Dastjerdi et al *(*1996), conforme abaixo:

- Controle de acesso – Deve garantir que todos os acessos diretos aos objetos do sistema devem ser regidos pelos privilégios e regras de acesso previamente definidos, ou seja, tem como principal objetivo prover diferentes tipos de acesso (Leitura, Escrita ou Execução) para um usuário ou grupo de usuários (*roles*). Isto pode acontecer também na forma de predicados (modificação da consulta) também chamado de controle de acesso granular;
- Auditoria – O objetivo é permitir o armazenamento de todos os acessos ao banco de dados para verificação posterior se alguma política de acesso ou autenticação foi violada;
- Autenticação – Identificar os usuários para garantir que cada nova conexão possui uma única pessoa associada. A Identificação do usuário é a base para cada mecanismo ou política de segurança, onde os usuários são habilitados para acessar os dados após se identificarem;
- Controle de Fluxo e Dedução – Garantir que dados de um determinado nível de segurança não possam ser transferidos para um nível de controle mais baixo e não permitir que um usuário seja capaz de deduzir informações via acesso indireto aos dados, ou seja, é preciso classificar a informação e protegê-la contra acessos não autorizados; e
- Consistência – Tem como objetivo definir o estado em que o banco de dados é considerado válido ou correto. Pode ser divida em três categorias:

- Integridade Operacional – Deve garantir a consistência lógica das transações no banco de dados mesmo durante a execução de transações concorrentes;
- Integridade Semântica – Tem como objetivo assegurar a consistência lógica dos dados modificados controlando os valores em um intervalo permitido, ou seja, garantir que o dado que está sendo inserido pertence ao tipo de dados definido na coluna; e
- Integridade Física – Determina que o banco de dados permita a sua reconstrução após paralisações não planejadas como, por exemplo, falta de energia.

2.3. Modelo de Autorização em Sistemas de Banco de Dados

Todo mecanismo de controle de acesso é baseado em algum modelo de autorização, definindo como o gerenciador do banco de dados deve implementar o controle de acesso aos dados (Sasaoka, 2002).

Um marco importante na história dos modelos de autorização foi definido por Griffiths e Wade, 1976. Batizado de System R, esse modelo introduziu conceitos básicos sobre a administração das autorizações de acesso bem como o controle das autorizações dependendo do conteúdo dos dados através de objetos do banco de dados chamados de visões.

A maioria dos modelos de autorização foram criados para os sistemas de bancos de dados em rede, hierárquicos e relacionais. Tais modelos trabalham, segundo Rabitti et al (1991) e Bertino et al (1997), com o seguinte conceito de autorização:

1. Autorização pode ser representada pela tupla <s, o, α>, onde:

   s ∈ S, sendo s um conjunto de sujeitos (usuários, grupos, entre outros) em um sistema;

   o ∈ O, sendo o um conjunto de objetos (tabelas, tuplas, por exemplo) em um sistema; e

   α ∈ A, sendo α os tipos de autorização (leitura, escrita, entre outros) em um sistema;
2. Uma função *f* pode ser definida para determinar se a Autorização (s, o, α) é verdadeira ou falsa:

$$f : S \times O \times A \rightarrow (True, False)$$

3. Dado o trio (s, o, α), se *f* (s, o, α) → True, então o sujeito *s* possui autorização do tipo α para acessar o objeto *o.*

Tal modelo foi utilizado como base pelos mecanismos de controle de acesso dos principais fornecedores de SGBD do mercado (Negrello e Wainer, 2006; Ferrari, 2010).

2.4. Mecanismos de Controle de Acesso

A proteção aos dados é garantida por diferentes componentes em um SGBD. O mecanismo de controle de acesso é o componente responsável pelo aspecto da confidencialidade dos dados (Bertino et al*,* 2005). O controle de acesso, nos bancos de dados relacionais, possui várias formas ou modelos de autorização. Conforme descrito por Dastjerdi et al (1996), os principais modelos utilizados para prover segurança em bancos de dados são:

- DAC (*Discretionary Access Control*) - Controle de Acesso Seletivo
- MAC (*Mandatory Access Control*) - Controle de Acesso Obrigatório; e
- RBAC (*Role Based Access Control*) - Controle de Acesso Baseado em Papéis;

2.4.1. Controle de Acesso Seletivo (DAC)

Mouelhi (2008) conceitua o controle de acesso DAC como um meio de restringir o acesso aos objetos baseado na identificação dos sujeitos e/ou grupos aos quais pertençam, ou seja, o modelo se baseia no conceito da propriedade dos objetos, onde existem os conceitos do proprietário (*owner*) e de recurso (um arquivo ou tabela, por exemplo). Desta forma, o proprietário do recurso pode escolher quem terá acesso ao recurso. Por este motivo, a política DAC é classificada como seletiva. O Quadro 2.1 destaca os tipos de acesso controlados pelo modelo DAC:

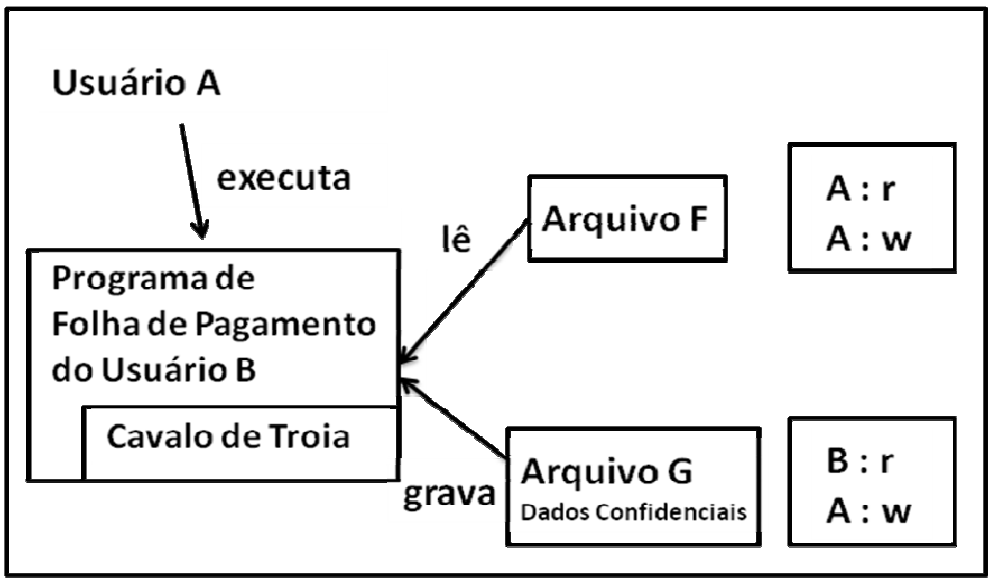
Quadro 2.1. Tipos de acesso controlados pelo modelo DAC (MOUELHI et al, 2008).

Permissão	Descrição
R	Permissão de leitura do objeto
W	Permissão de escrita do objeto
X	Permissão de execução do objeto
C	Permissão de controle do objeto, ou seja, permite modificar as permissões R, W ou X do objeto.
CP	Permissão de controle e de repassar permissões ou controle.

O modelo DAC permite visualizar uma matriz de permissões na qual se tem os proprietários e recursos como linhas e colunas da referida matriz. A interseção entre o proprietário e o recurso indica quais permissões estão habilitadas. Um exemplo de controle de acesso seletivo é o utilizado pelo sistema operacional UNIX (Dodonov et al, 2004; Ausanka-Crues, 2006) no qual as permissões de acesso são armazenadas em cada recurso (arquivo, por exemplo).

O modelo DAC ainda é bastante utilizado e difundido em sistemas comerciais como os sistemas operacionais UNIX e algumas versões do Windows (Dodonov et al, 2004; Ausanka-Crues, 2006). No entanto, permitir que usuários controlem o acesso aos dados possibilita o surgimento de programas do tipo "Cavalo de Tróia". Isto acontece quando, por exemplo, o usuário A que possui acesso a dados de folha de pagamento, permite a cópia (não autorizada) desses dados ao executar um programa que recebeu privilégio do usuário B. O programa criado pelo usuário B, além de desempenhar suas funções originais, pode copiar inadvertidamente dados confidenciais aos quais teve acesso pela execução do programa pelo usuário A. A Figura 2.2, adaptada dos estudos de Vinod (2008), ilustra como funciona o mecanismo de ataque do tipo "Cavalo de Tróia" descrito.

Figura 2.2. Mecanismo de ataque do tipo cavalo de Tróia



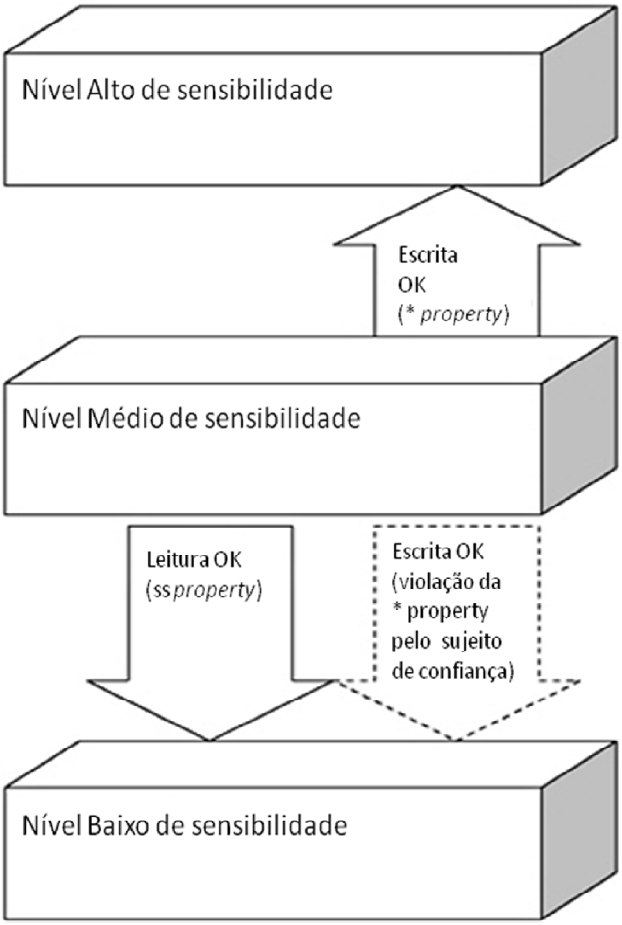
Fonte: Adaptado de Vinod, 2008.

2.4.2. Controle de Acesso Mandatório (MAC)

Diferentemente do modelo DAC, o modelo de controle de acesso mandatório (MAC) garante que o fluxo da informação aconteça de acordo com a política de segurança definida, pois o MAC só permite que os usuários acessem as informações do nível de segurança ao qual o usuário está designado (Dastjerdi et al, 1996).

Também conhecido como modelo multinível ou hierárquico, o MAC satisfaz aos requisitos de organizações naturalmente hierárquicas como as das áreas militar e governamental. Tal modelo é baseado na classificação da informação em níveis de segurança. Para exemplificar esse modelo, seja a política de segurança definida no modelo criado por Bell-Lapadula (Rushby, 1984), no qual a informação é classificada em "Top Secret", "Secret", Confidential" e "Unclassified" e cuja ideia central pode ser visualizada na Figura 2.3.

Figura 2.3. Modelo Bell-Lapadula



Fonte: Krutz e Vines, 2007.

A Figura 2.3 demonstra que tal modelo assegura que informações de um determinado nível não fluam para um nível de segurança inferior, a partir de dois princípios:

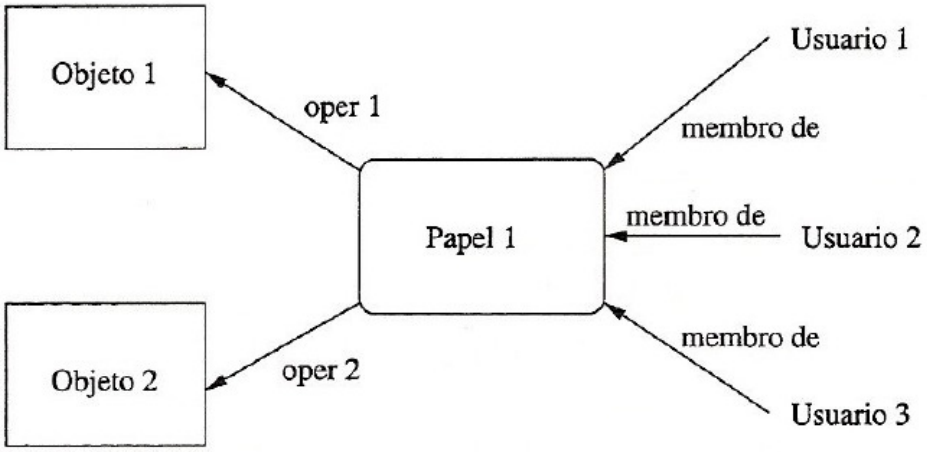
- ***Simple-Security Property***: O sujeito pode ler um objeto somente se o nível do objeto for <= ao nível em que o sujeito se encontra; e
- ****-Property (star-property):*** O sujeito não pode escrever em objetos de um nível inferior ao seu.

O segundo princípio protege o modelo MAC contra invasões do tipo "cavalo de Tróia", permitidas pelo modelo DAC. No entanto, o modelo MAC é bastante complexo e possui alto custo para ser cumprido, pois necessita que as aplicações sejam ajustadas para os níveis criados (Ausanka-Crues, 2006).

2.4.3. Controle de Acesso Baseado em Papéis (RBAC)

A política de controle de acesso baseado em papéis (RBAC) foi motivada pela necessidade de simplificar a administração das políticas de controle de acesso (Bertino et al, 2005). Nesse modelo, a autorização do acesso aos recursos é realizada a partir das funções desempenhadas pelo usuário dentro da organização. A partir desta premissa, foi criado o conceito *role* (papel), o qual é uma camada de abstração para grupos nomeados de privilégios que podem ser concedidos ao usuário conforme pode ser observado na Figura 2.4.

Figura 2.4: Relacionamento entre usuários, papéis, operações e objetos.



Fonte: Sasaoka, 2002.

Ferraiolo e Huhn (1992) apresentam a descrição formal do RBAC composta de três regras básicas, descritas abaixo:

1. Associação do Papel - Um sujeito só poderá executar uma transação se estiver associado a uma *role* (papel), ou seja, um usuário deve possuir um papel ativo para realizar uma transação. A ativação deste papel geralmente é realizada quando o usuário realiza a autenticação no sistema;
2. Autorização do Papel - Os papéis ativos para o sujeito devem ser autorizados para o referido sujeito. Esta regra garante que os usuários só podem utilizar os papéis que foram autorizados para uso; e
3. Autorização da Transação - Um sujeito só pode executar uma transação se esta foi autorizada para o papel ativo do sujeito.

Grandes fornecedores de software aderiram ao modelo de controle de acesso baseado em papéis (RBAC), tais como SAP (SAP Website, 2012), Oracle (Oracle Website, 2012), PostgreSQL (PostgreSQL, 2012), Solaris (Oracle Solaris 11, 2012) e SELinux (Security-Enhanced Linux, 2012) já dão suporte ao referido modelo. No entanto, em grandes sistemas nos quais o número de usuários, objetos, herança e papéis e, principalmente, a necessidade de controle de acessos específicos é alta, torna o trabalho de organização e administração pesado (Ausanka-Crues, 2006). O Quadro 2.2 exibe uma análise comparativa entre os mecanismos de controle de acesso abordados:

Quadro 2.2. Análise comparativa entre os principais mecanismos de controle de acesso.

Comparativo entre os Mecanismos de Controle de Acesso		
DAC	MAC	RBAC
Simples	Complexo	Simples
Baixo Custo	Alto Custo	Baixo Custo
Dificuldade de Administração	Dificuldade de Administração	Facilidade de Administração
Baseado no conceito da propriedade dos objetos	Baseado na classificação da informação em níveis de segurança	Baseado em funções (papéis) desempenhadas pelo usuário dentro da organização
Matriz de Permissões	Modelo Hierárquico	Relacionamento entre usuários, papéis, operações e objetos
Permite Cavalo de Tróia	Não Permite Cavalo de Tróia	Não Permite Cavalo de Tróia

2.5. Considerações Finais

Neste capítulo, foram abordados os principais conceitos sobre segurança de informação. Verificou-se que a tríade da CIA (confidencialidade, disponibilidade e integridade) é bastante abrangente e garante a segurança de uma variedade de sistemas computacionais, dentre eles os sistemas de banco de dados. Foram citados também os mecanismos ou requerimentos disponíveis para permitir a segurança dos sistemas de banco de dados.

Os princípios básicos de um modelo de autorização bem como os principais mecanismos de controle de acesso utilizados pelos sistemas gerenciadores de banco de dados hierárquicos, em rede e relacionais foram destacados.

Nesse contexto, o requerimento chamado "controle de acesso", derivado do conceito de confidencialidade da segurança de informações, foi escolhido como área principal de aprofundamento no decorrer dos próximos capítulos, pois o controle de acesso granular, principal objeto deste estudo, é um tipo especial de controle de acesso. No próximo capítulo será focalizado o conceito e principais mecanismos para prover o controle de acesso granular.

Capítulo 3: Controle de Acesso Granular aos Dados

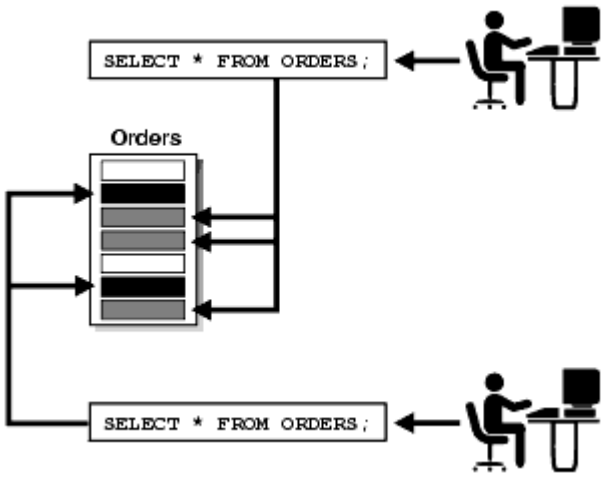
O conceito de controle de acesso granular aos dados foi criado por Stonebraker e Wong (1974) como parte de um sistema de controle de acesso para o sistema de banco de dados INGRES. A ideia central da solução foi baseada na tecnologia da modificação transparente e dinâmica da consulta antes que os comandos fossem processados pelo INGRES para garantir que os usuários só tivessem acesso aos dados autorizados para cada usuário específico (Bertino et al., 2005; Wang et al., 2007; Shi e Zhu, 2010).

Lindblad (2001) definiu granularidade fina de acesso a dados como sendo a modificação dinâmica da consulta para forçar políticas de segurança nos objetos do banco de dados que possuam uma política associada. Em outras palavras, controle de acesso granular aos dados é a habilidade de se adicionar dinamicamente um predicado (cláusula WHERE) em qualquer tabela ou visão do banco de dados.

A palavra consulta na citação acima se refere tanto a instruções SQL SELECT quanto a qualquer comando de Linguagem de Manipulação de Dados (DML), ou seja, INSERT, UPDATE e DELETE.

A Figura 3.1 mostra um cenário simples de controle de acesso granular, onde dois usuários distintos executam o mesmo comando e obtêm resultados diferentes.

Figura 3.1 – Exemplo de Controle de Acesso Granular



Fonte: Oracle Label Security Administrator's Guide, 2012.

O controle de acesso fino (granular) aos dados pode ser criado nos seguintes níveis:

- Camada de aplicação;
- Utilizando Visões dentro do Banco de Dados; e
- Utilizando características próprias de um SGBD específico.

As principais características são detalhadas nas subseções 3.1 a 3.3, bem como as vantagens e desvantagens de cada um dos níveis listados nas referidas subseções.

3.1. Controle de Acesso Granular na Camada de Aplicação

O controle de acesso granular aos dados é tradicionalmente implementado dentro da camada da aplicação, sem função direta associada ao banco de dados. Surajit Chaudhuri et al. (2007) destacam várias desvantagens em utilizar a implementação do controle de acesso granular através da aplicação. As principais são:

I. O controle de acesso está geralmente distribuído em muitas linhas de código, o que requer esforço de manutenção e aumenta as chances de problemas de segurança originadas por programadores. Em contraste, prover o suporte para granularidade fina no acesso aos dados pelo engenho do sistema de banco de dados determina que as políticas de acesso sejam uniformes e aplicadas a todos os acessos; e

II. As aplicações geralmente se conectam ao banco de dados utilizando *login* único. Fazendo analogia a um sistema operacional, todas as consultas são executadas com privilégio do usuário administrador. Como a área a ser protegida é ampla, o potencial dano ocasionado por acessos maliciosos é alto e podem provocar danos irreparáveis aos dados do banco de dados.

3.2. Controle de Acesso Granular no nível do SGBD

Em várias aplicações os mecanismos de segurança aos dados são introduzidos em um módulo à parte dentro da própria aplicação. Porém, alguns conceitos de segurança que são nativos dos sistemas de banco de dados também são utilizados para prover o controle no acesso aos dados das organizações. O modelo mais tradicional e conhecido de controle de acesso nos bancos relacionais é o RBAC descrito na seção 2.4.3. Tal modelo se baseia no conceito

de administrar as permissões dos usuários de acordo com os papéis desempenhados (Ferraiolo et al, 1995). Em termos práticos, os sistemas de banco de dados relacionais, em sua maioria, utilizam os seguintes mecanismos para assegurar o controle de acesso aos dados:

- Permissões (*Grant*) – Permitem conceder ou revogar o acesso a um ou mais objetos do banco de dados. Cada objeto possui um determinado conjunto de privilégios concedíveis como, por exemplo, uma tabela pode permitir a inclusão, alteração, exclusão ou a seleção (em uma ou mais colunas);
- Papéis (*Roles*) – Objetos capazes de encapsular grupos de privilégios;
- Visões (*Views*) – Utilizadas para agrupar e filtrar dados no nível de coluna e linha; e
- Procedimentos Armazenados (*Stored Procedures*) – Fornecem uma camada de aplicação onde o usuário só tem acesso aos métodos e não aos dados.

No entanto, o modelo RBAC (*grants* e *roles*) e procedimentos armazenados por si só não permitem o controle de acesso granular aos dados. As visões, por sua vez, permitem tal controle e são bastante utilizadas por proporcionarem segurança não nível de coluna e de linha. Segundo Agrawal et al (2005), este método é útil quando o número de restrições de acesso é pequeno ou quando a identificação dos grupos de usuários é uma tarefa fácil. Caso tais premissas não sejam verdadeiras, a utilização de visões pode tornar a manutenção delas uma tarefa difícil, visto que o tamanho do código fonte da visão crescerá proporcionalmente ao número de grupos de usuários e de restrições existentes no sistema.

Os estudos de Kabra et al (2006) demonstram a utilização de visões para prover acesso granular aos dados a partir de um exemplo que considera uma tabela chamada employee (emp_id, name, dept_id, salary). A partir desta tabela, que armazena informações dos funcionários de uma empresa, é necessário controlar o acesso para que os usuários só consigam acessar os funcionários que pertençam a um conjunto conhecido e definido de departamentos representado pelo conjunto Q1. A partir desta regra, é possível criar uma política de segurança através de uma visão com o Código 3.1.

Código 3.1: Criação de visão para controlar o acesso granular na tabela employee.

```
CREATE VIEW myemployees AS
SELECT *
FROM employee
WHERE dept_id in Q1
```

A utilização de visões para prover o controle de acesso granular aos dados é muito simples, como visto no exemplo anterior. Porém, esta alternativa possui algumas limitações que tornam esta solução não recomendada para a maioria dos casos. Opyrchal et al (2011) descrevem algumas limitações desta abordagem:

I. Uma *view* deve ser criada para cada entidade com restrição de acesso diferente. Dependendo da aplicação, esta limitação pode ser resolvida com tabelas ou colunas que contenham indicadores que permitam o filtro de acesso de maneira mais genérica;
II. É necessário modificar a aplicação para acessar os dados pelas *views* e não pelas tabelas/*views* originais;
III. A segurança dos dados não é total, caso os usuários possam acessar as tabelas que são utilizadas como base para a criação das visões;
IV. As lógicas das visões podem se tornar bastante complexas para garantir o controle de acesso, dificultando o trabalho de manutenção, pois fica difícil distinguir qual parte da lógica do SELECT da visão que corresponde à segurança e qual corresponde à própria definição da visão; e
V. Quanto maior a quantidade de perfis de usuários que acessam o sistema, maior será o número de visões, maior será o número de objetos do banco de dados e maior será o custo de manutenção na alteração de qualquer regra de segurança que envolva mais que uma visão.

3.3. Mecanismos de Controle Granular de SGBD do Mercado

O controle de acesso granular aos dados já é comercializado por alguns dos grandes fornecedores de SGBD. O Oracle (Oracle Website, 2012) possui os produtos Virtual Private Database (VPD) (Oracle Database Security Guide, 2011) e o Label Security (Oracle Label Security Administrator's Guide, 2011), o Sybase (Sybase Corporation, 2012) disponibiliza o Row-Level Authorization (Sybase Technical Library, 2012) e o DB2 (DB2 Database Software Website, 2012) da IBM (IBM Corporation, 2012) criou o Multiple Level Security (Bird, 2000). Já no SQL Server 2005 (Microsoft SQL Server 2005 Website, 2012), da Microsoft (Microsoft Corporation, 2012), o mecanismo para utilizar a segurança ao nível de linha continua sendo as Visões. Nos produtos da Oracle e da Sybase, o modelo de autorização consiste na mudança dinâmica da consulta, com a adição de predicados aos comandos executados. Já no DB2, para cada política de segurança, um objeto do tipo visão (*view*) é criado com a restrição definida pela política. Então, é dado acesso para a visão ao usuário.

Os principais mecanismos disponibilizados pelo mercado para realizar o controle de acesso granular aos dados são descritas nas subseções seguintes. Três grandes fabricantes foram escolhidos para este estudo em função das características peculiares de suas soluções.

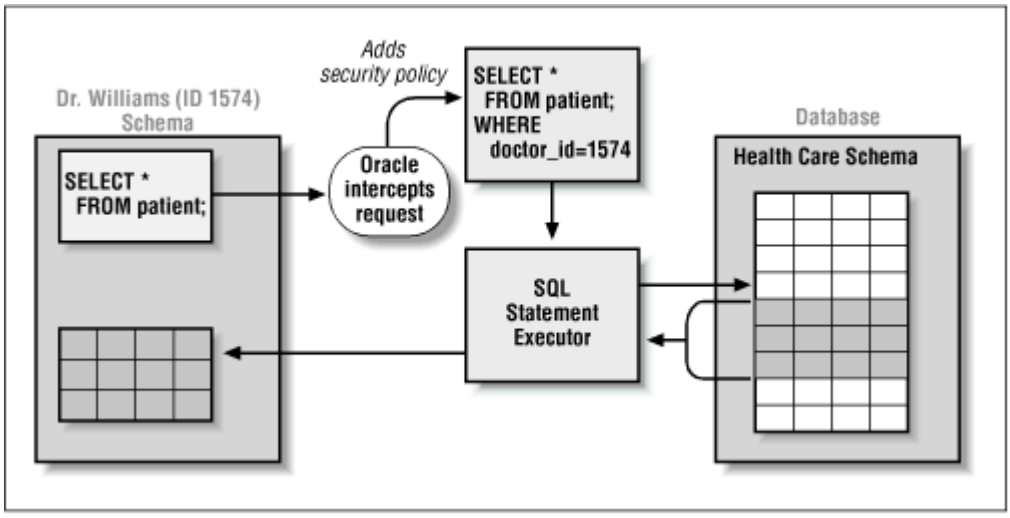
3.3.1. Virtual Private Database / Label Security (Oracle)

O VPD (Virtual Private Database) foi implementado pela Oracle Corporation desde o ano de 1999, na versão 8i do sistema de banco de dados Oracle. No entanto, esse recurso só está disponível para quem estiver disposto a adquirir a licença da versão Enterprise Edition (Oracle Corporation, 2011).

Os predicados (cláusula WHERE) são retornados por funções que aplicam a segurança imposta pela política de acesso ao objeto. Tais funções podem realizar chamadas a qualquer outro tipo de procedimento, incluindo procedimentos externos escritos na linguagem C (Ritchie, 2011) ou Java Stored Procedures (Oracle Corporation, 2011), que são procedimentos escritos na linguagem Java e armazenados no SGBD ORACLE. Com essa flexibilidade, é possível buscar informações do sistema operacional, de um arquivo específico ou até de uma unidade de armazenamento que contenha as regras (políticas) de acesso. Feuerstein (1999) descreve, por meio da Figura

3.2, o fluxo que ocorre para acontecer o mecanismo de modificação dinâmica das consultas.

Figura 3.2: Mecanismo de modificação dinâmica da consulta do VPD.



Fonte: Feuerstein, 1999.

As políticas de acesso referem-se tanto a instruções SQL SELECT quanto a qualquer comando de Linguagem de Manipulação de Dados (DML - Data Manipulation Language), ou seja, INSERT, UPDATE ou DELETE.

Pelo fato de ser forçado pelo SGBD ORACLE, qualquer tentativa de acesso indireto ao objeto também sofrerá mudança dinâmica no acesso, de acordo com a política de acesso estabelecida. Como exemplos de tentativas de acesso indireto a objetos, pode-se citar:

- Visões;
- Procedimentos Armazenados (*Procedures*, *Functions* e *Packages*); e
- Relatórios desenvolvidos para uma aplicação.

Uma Política de Segurança é representada no Virtual Private Database através de uma função armazenada no banco de dados onde esta função tem o papel de retornar o predicado que será anexado à cláusula WHERE do comando executado, sendo assim, utilizado como filtro de dados dinâmico no momento da execução da consulta. Geralmente, esta função utiliza os valores atribuídos às variáveis de sessão pelos contextos de aplicação (conceito de variáveis globais) no momento da conexão do

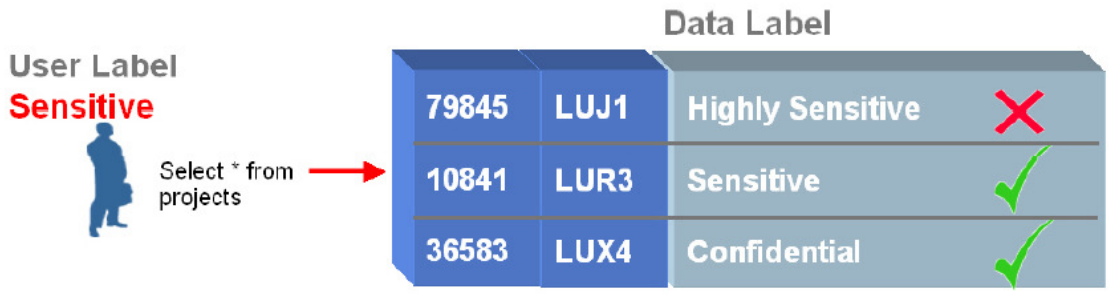
usuário no banco de dados para determinar "o que" o usuário conectado poderá ter acesso dentro do sistema.

Para que haja a modificação dinâmica na consulta em determinada tabela, é necessário associar a função que define o predicado a ser anexado à tabela que se deseja implementar a política de segurança (ver Anexo A). Para realizar esta associação pode-se utilizar um pacote chamado DBMS_RLS que é criado no momento da criação do banco de dados. O pacote DBMS_RLS consiste em quatro métodos (procedimentos) principais (Oracle Corporation, 2008):

1. ADD_POLICY – Adiciona uma política a uma tabela, visão ou sinônimo;
2. DROP_POLICY – Remove uma política de uma tabela, visão ou sinônimo;
3. ENABLE_POLICY – Habilita ou desabilita uma política já existente; e
4. REFRESH_POLICY – Força uma nova análise dos cursores abertos associados à política a aproveitarem imediatamente uma política nova ou alterada.

O Oracle também oferece, em licenciamento à parte, para os clientes que já adquiriram a versão Enterprise do SGBD, um mecanismo chamado Label Security para controle de acesso granular aos dados. O Oracle Label Security é utilizado para controlar o acesso às linhas das tabelas de um banco de dados rotulando, de maneira similar ao controle de acesso MAC (*Mandatory Access Control*), cada linha e cada usuário e concedendo ou negando o acesso de acordo com esses rótulos, conforme o exemplo mostrado na Figura 3.3.

Figura 3.3: Exemplo de modificação dinâmica da consulta do Label Security



Fonte: Oracle Corporation, 2009.

Conforme Azevedo et al (2010), os rótulos são compostos por três componentes: um nível, um ou mais compartimentos e um ou mais grupos. O nível representa a sensibilidade dos dados e pode ser, por exemplo, público, corporativo, secreto, entre outros. Os compartimentos, que podem também ser chamados de categorias, organizam as informações horizontalmente. A definição de compartimento para um rótulo é opcional. Por fim, os grupos são geralmente utilizados para representar o controle de acesso organizacional, pois podem ter uma hierarquia. A definição de grupos para um rótulo é opcional. O Quadro 3.1 mostra exemplos práticos de definição para níveis (*Levels*), compartimentos (*Compartments*) e grupos (*Groups*) em sistemas Governamentais (Militares), Justiça, Recursos Humanos e de Saúde (Oracle Corporation, 2009).

Quadro 3.1- Exemplo de níveis, compartimentos e grupos por tipo de negócio (Oracle Corporation, 2009).

INDUSTRY	LEVEL	COMPARTMENT	GROUP
Government and Defense	Confidential Secret Top Secret	Alpha Beta	NATO Homeland Security
Law Enforcement	Level 1 Level 2 Level 3	Border Security Drug Enforcement	Local Jurisdiction FBI Justice Department
Human Resources	Confidential Sensitive Highly Sensitive	PII Data Investigation	Global United States Europe
Health Care	Confidential Public	VIP Controls	Physician Laboratory

O Label Security utiliza internamente o mecanismo do Virtual Private Database para garantir o acesso aos dados sensíveis. Conforme descrito por Ron Ben-Natan, (2005), o mecanismo de controle do Label Security pode ser visualizado na Figura 3.4.

Figura 3.4: Mecanismo de modificação dinâmica da consulta do Label Security

Fonte: Oracle Corporation, 2009.

O Quadro 3.2 descreve as vantagens e desvantagens encontradas no conceito e criação de políticas de segurança com a tecnologia do Virtual Private Database.

Quadro 3.2. Vantagens e Desvantagens do Virtual Private Database (Marques, 2005).

VANTAGENS	DESVANTAGENS
Simplifica o Desenvolvimento da Aplicação – Separa controle de acesso da aplicação e o coloca junto dos dados	**Difícil Manutenção**. Não existe ferramenta para controle das políticas, ou seja, qualquer nova política ou manutenção de uma política existente requer uma nova programação.
Garante que os dados no banco de dados estão sempre protegidos. Não importa qual a ferramenta de acesso os dados, tem-se certeza que a política de segurança será chamada e não pode ser burlada.	**Disponível apenas na versão do Oracle Enterprise Edition**. Para utilizar este recurso no Oracle, o custo de aquisição custa a partir de U$ 47.500,00 (Oracle Corporation, 2012).
Permite alterações de evolução nas políticas de segurança sem impacto no lado do cliente da aplicação	**É difícil de rastrear erros.** Como o Controle de acesso de granularidade fina acontece em background com a modificação dinâmica da consulta ajustar erros em tempo de execução não é uma tarefa fácil.
Simplifica o gerenciamento dos objetos do banco de dados. Reduz o número total de objetos necessários para dar suporte para a aplicação.	-
Possui bom desempenho. A utilização de contextos de aplicação permite a utilização e traz os benefícios do SQL compartilhado.	-

3.3.2. Row-Level Authorization (Sybase)

O mecanismo de segurança do Sybase ASE (Adaptive Server Enterprise) a partir da versão 12.5 permite criar políticas de segurança onde os predicados são associados aos campos das tabelas. Diferentes políticas podem ser criadas em diferentes colunas e podem ser combinadas utilizando os operadores lógicos OR ou AND definidos pela política de segurança e são adicionados na cláusula WHERE da consulta (Chaudhuri et al., 2007).

Esta funcionalidade é chamada de controle de acesso no nível de linha, do inglês *"row level access control"* e permite restringir o acesso às linhas das tabelas definindo as regras de acesso e ligar tais regras às tabelas. O acesso a dados dinâmicos (ex: usuário conectado no momento) podem ser controlados criando objetos chamados contextos de aplicação e gatilhos de *login* (Agrawal et al, 2005). A sintaxe para habilitar o controle de acesso ao nível de linha no servidor de banco de dados é descrita no Código 3.2.

Código 3.2. Comando para habilitar o controle de acesso ao nível de linha no Sybase.

```
sp_configure "enable row level access", 1
```

Habilitar esta opção pode provocar um leve aumento no consumo de memória no Sysbase (Sybase Technical Library, 2005). Além disso, é necessário adquirir uma licença chamada ASE_ASM para uso deste mecanismo de segurança. O mecanismo de utilização do controle de acesso no nível de linha no Sybase pode ser realizado (ver Anexo B) em alguns passos descritos por Garbus et al (2002). A partir destes passos, é possível verificar que a utilização do controle de acesso no nível de linha é bem simples de ser criado. No entanto, este modelo não permite autorização de acesso por coluna. Além disso, não existem ferramentas para simplificar a administração, tais como gerenciar grupos de usuários e as autorizações dos respectivos grupos.

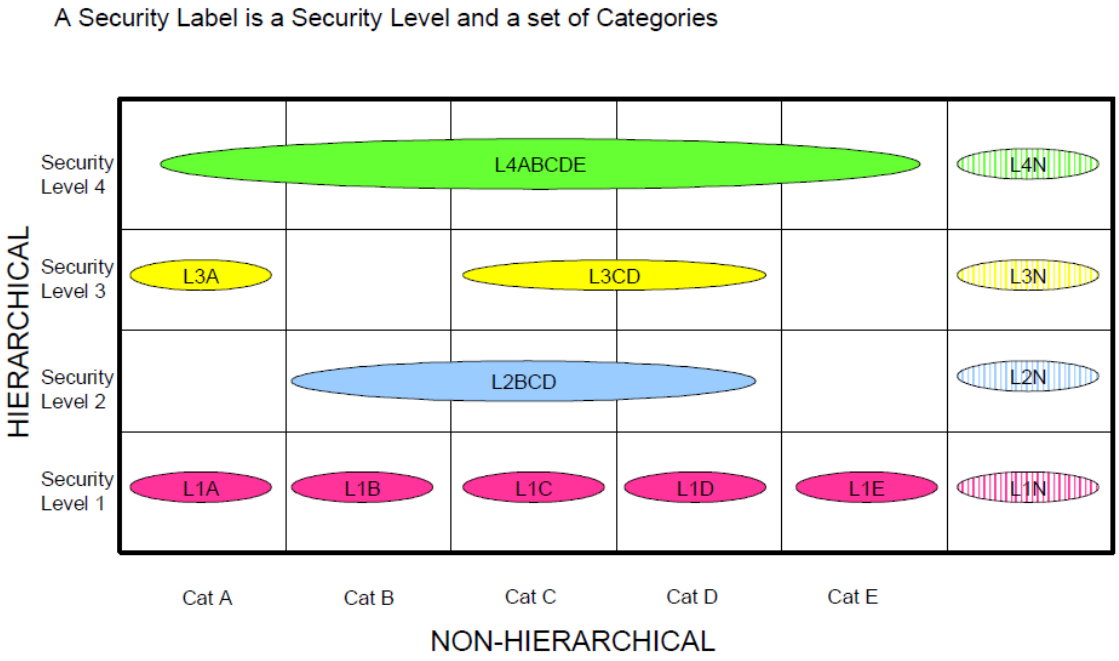
3.3.3. Multiple Level Security (DB2)

A partir do sistema de banco de dados DB2 V8, da IBM, é possível controlar a segurança dos dados no nível de linha a partir de um mecanismo chamado MLS (Multi-Level Security) (Bergh, 2006). No entanto, este mecanismo só funciona em conjunto com um produto de gerenciamento de segurança como, por exemplo, o RACF (Resource Access Control Facility, 2012) cujo licenciamento é independente do SGBD. Este produto é um serviço de segurança que autoriza acesso do usuário a recursos protegidos. Foi originalmente desenvolvido pela IBM (IBM Corporation, 2012) em 1976 e ainda é utilizado até os dias de hoje para gerenciar a segurança em *mainframes*. Ao longo dos anos foi aprimorado e retirado do *mainframe* para outros ambientes. O RACF gerencia autenticação de usuários, acesso a dados, criptografia e várias outras funcionalidades de segurança (Ron Ben-Natan, 2005).

Antes de explicar o mecanismo de controle no acesso granular aos dados é preciso definir alguns conceitos importantes utilizados no mecanismo desenvolvido pela IBM. De acordo com Rayns et al (2005), os conceitos abaixo são primordiais para o entendimento da solução:

- Objeto – recurso do sistema que deve ser controlado. Para o DB2, por exemplo, objetos seriam as tabelas e linhas do banco de dados;
- Assunto – Entidade que precisa acessar os objetos. Exemplo: usuário do banco de dados;
- Rótulo de Segurança (SECLABEL) – Indica o nível de hierarquia ou classificação da informação, ou seja, é uma combinação do nível de segurança hierárquica da informação com um ou mais categorias não hierárquicas. Na Figura 3.5 é visualizada uma tabela básica com níveis de hierarquia de 1 a 4 e categorias não hierárquicas de A até E. O objetivo da tabela é criar rótulos de segurança unindo um nível de segurança e zero ou mais categorias.

Figura 3.5: Exemplo de definição dos níveis de segurança no MLS



Fonte: Bergh, 2006.

A partir da Figura 3.5 pode-se exemplificar alguns rótulos de segurança verificando a interseção entre os níveis de segurança e as categorias:

- L1A é a combinação do nível de segurança 1 com a categoria A;
- L2BCD é a combinação do nível de segurança 2 com as categorias B, C, e D; e
- L3N é a combinação do nível de segurança 3 sem categorias associadas.

Para ativar o mecanismo de autorização no banco de dados, é necessário adicionar uma coluna especial para atuar como rótulo de segurança. O valor contido nesta coluna é comparado com o valor contido no gerenciador de segurança (RACF).

O mecanismo de utilização do controle de acesso no nível de linha no DB2 pode ser realizado (ver Anexo C) em alguns passos descritos por Bergh (2006).

3.3.4. Comparativo entre os Mecanismos

Os mecanismos descritos nesta seção são proprietários, pertencentes a alguns dos principais fornecedores de sistemas de banco de dados no mercado, ou seja, alguns dos conceitos e características apresentados em uma solução não estão nas outras. Além disso, mesmo quando o conceito é o mesmo, a forma de realizar o controle de acesso granular e as respectivas ferramentas para definição, controle e monitoramento do controle de acesso são diferentes.

O objetivo desta seção é apresentar um quadro comparativo entre as soluções analisadas para o controle de acesso granular. Os campos exibidos no Quadro 3.3 e as características comparadas são as seguintes:

- Licenciamento – Indica se deverá ser pago algum valor para utilizar o mecanismo de segurança de acesso granular além do valor pago pela utilização do SGBD;
- Controle de acesso por coluna – Indica se o mecanismo de controle criptografa ou suprime colunas com conteúdo protegido;
- Controle de acesso por linha - Indica se o mecanismo de controle suprime linhas com conteúdo protegido;
- Alteração do modelo de dados – Indica se a solução necessita que o modelo de dados atual seja alterado adicionando colunas que servirão como indicador do nível de segurança de uma linha;
- Múltiplas políticas por tabela – Indica se o mecanismo de controle de acesso permite mais de uma política por objeto (tabela) ao mesmo tempo, adicionando predicados; e
- Solução única para vários SGBD – Indica se a ferramenta proprietária de um sistema gerenciador de banco de dados pode ser utilizada por outro SGBD, como um mecanismo independente.

Quadro 3.3: Quadro comparativo entre os fornecedores e as características disponíveis em relação ao controle de acesso granular.

Fornecedor / Característica	Oracle 11g	Sybase	DB2
Licenciamento à parte	Não *	Sim	Sim
Controle de acesso por coluna	Não	Não	Não
Controle de acesso por linha	Sim	Sim	Sim
Alteração do modelo de dados	Não	Não	Sim
Múltiplas políticas por tabela	Sim	Sim	Sim
Solução única para vários SGBD	Não	Não	Não

* Só está disponível na versão Enterprise Edition.

O quadro comparativo evidencia deficiências entre todas as soluções como o "Controle de acesso por coluna" e "Solução única para vários SGBD" onde nenhum dos fornecedores analisados disponibiliza. Além disso, fica claro que a questão do licenciamento é um limitador para aquisição e utilização das soluções disponíveis no mercado.

3.3.5. Considerações Finais

Neste capítulo, foi contextualizado o assunto de segurança das informações e descritos, de maneira breve, os principais mecanismos utilizados no mercado para controle de acesso granular aos dados dos bancos de dados. Por fim, foram comparados tais mecanismos, destacando-se as vantagens e desvantagens de cada implementação.

No próximo capítulo será apresentado o estado da arte no que tange ao processo de controle de acesso granular aos dados. Serão mostradas também algumas metodologias propostas. Será realizada ainda uma análise a respeito das soluções abordadas. Por fim, será exposto um comparativo entre as soluções abordadas e a proposta deste trabalho de uma solução para controlar o acesso granular aos dados.

Capítulo 4: Metodologias para Controle de Acesso Granular aos Dados

As pesquisas relacionadas ao controle de acesso granular nos sistemas de bancos de dados foram iniciadas na década de 70 (Opyrchal et al, 2011). No entanto, o interesse dos principais fornecedores de sistemas de banco de dados do mercado só despertou a partir do final da década de 90 com o lançamento do Oracle 8i (Oracle8i Database Online Documentation, 2012) no ano de 2000 e posteriormente do Sybase ASE 12.5 (Contents of the Adaptive Server Enterprise 12.5 Collection, 2012) em 2001.

Na literatura vários trabalhos têm sido sugeridos sobre o tema controle de acesso granular aos dados. Algumas dessas propostas tentam contribuir em caráter conceitual, ou seja, sugestões de melhoria do controle de acesso disponibilizado atualmente pelos principais fornecedores de SGBD do mercado com o objetivo de facilitar ou prover mecanismos de controle de acesso granular (Rizvi et al, 2004; Purevjii et al, 2007; Olson et al, 2008; Halder e Cortesi, 2010).

Outras propostas tentam fornecer um mecanismo "prático" com base em *plugins* ou aplicativos preparados para modificar dinamicamente a consulta com o objetivo de controlar o acesso de maneira independente do fornecedor do SGBD (Shi e Zhu, 2010; Purevjii et al, 2007; Vinod, 2008). Nas subseções seguintes serão detalhados alguns estudos importantes de cunho conceitual e prático com o objetivo de analisar a situação atual em relação ao problema alvo bem como comparar as metodologias empregadas nas respectivas pesquisas com a metodologia proposta neste trabalho.

4.1 Estendendo Sistemas de Banco de Dados Relacionais para aplicar Políticas de Privacidade automaticamente

Agrawal et al (2005) propõem extensões SQL no contexto da privacidade dos dados. Esse artigo serviu de base para o artigo que será descrito na seção 4.2 e se propõe a construir uma linguagem, chamada "Fine grained restriction syntax" para controlar o acesso granular aos dados. Abaixo é mostrada a sintaxe da referida linguagem:

Código 4.1. Sintaxe da extensão SQL para controle granular aos dados na proposta de Agrawal et al (2005).

Create restriction restriction-name
on table-x
for auth-name-1 [**except** auth-name-2]
(((**to columns** column-name-list)
| (**to rows** [**where** search-condition])
| (**to cells** (column-name-list [**where** search-condition])+)
)
[**for purpose** purpose-list]
[**for recipient** recipient-list]
)+
command-restriction

Na situação descrita para o artigo anterior, onde usuário só teria acesso aos seus dados na tabela EMPLOYEE, e não a todos os dados da tabela, a restrição poderia ser criada a partir do comando abaixo:

Código 4.2. Exemplo de comando para controle granular aos dados na proposta de Agrawal et al (2005).

Create restriction r1
on Employee
for public
to rows where name = **user**
restricting access to all

Uma restrição pode ser especificada no nível de coluna, linha ou célula. Além disso, mais de uma restrição pode ser criada para uma determinada tabela do mesmo usuário. Abaixo são listados exemplos da sintaxe na definição de autorização para cada um dos níveis citados:

- Nível de coluna – Especifica um conjunto de colunas que um usuário pode acessar em uma determinada tabela. Abaixo é listado um exemplo de criação da

restrição r1 que limita o usuário Bob a acessar apenas a coluna id da tabela *Customer*:

Código 4.3. Exemplo de comando para controle granular aos dados no nível de coluna na proposta de Agrawal et al (2005).

```
create restriction r1
on Customer
for user Bob
to columns id
restricting access to all
```

- Nível de linha – Especifica um conjunto de linhas que um usuário pode visualizar em uma determinada tabela. Abaixo é listado um exemplo de criação da restrição r3 que limita todos os usuários a acessar apenas os registros da tabela *Customer* cuja coluna *name* coincida com o usuário conectado com o banco de dados:

Código 4.4. Exemplo de comando para controle granular aos dados no nível de linha na proposta de Agrawal et al (2005).

```
create restriction r3
on Customer
for public
to rows where name = user
restricting access to all
```

- Nível de célula – Especifica um conjunto de linhas e colunas que um usuário pode visualizar em uma determinada tabela. Abaixo é listado um exemplo de criação da restrição r4 que limita o usuário Bob a acessar, na tabela *Customer*, informações do nome do cliente (coluna *name*) e o telefone (coluna *phone*) apenas se o cliente liberou o acesso a tal informação:

Código 4.5. Exemplo de comando para controle granular aos dados no nível de célula (linha e coluna) na proposta de Agrawal et al (2005).

```
create restriction r4
on Customer for user Bob,
to cells name,
(phone where exists (
select 1
from SysCat.Choices Customer c
where c.ID = Customer.ID and c.C1 = 1))
for purpose marketing
for recipient others
restricting access to select
```

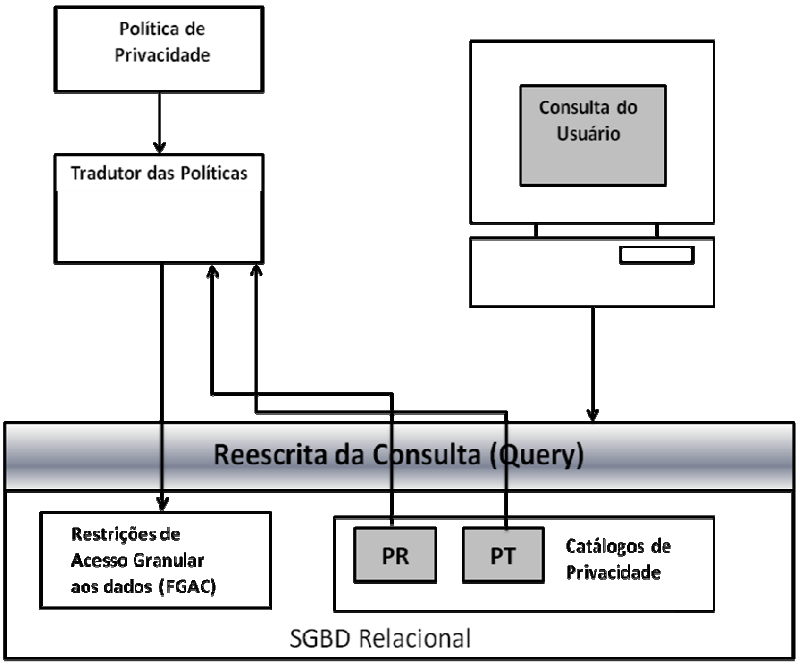
A última linha do comando acima indica que a restrição de acesso atua apenas em comandos *select*. No entanto, é possível restringir o acesso em qualquer combinação dos comandos *selec*t, *delete*, *insert*, ou *update* a partir da sintaxe abaixo:

Código 4.6. Exemplo de comando para restringir o controle granular aos dados na proposta de Agrawal et al (2005).

```
restricting access to (all | (select | delete | insert | update)+ )
```

O aspecto teórico desse artigo também é bastante rico e interessante, inclusive mostra como seria o algoritmo para controlar restrições utilizando um Hippocratic database system (Agrawal et al, 2002), que são ideias e definições (não implementadas) de bancos de dados que devem atender questões fundamentais de privacidade das informações, onde a eficiência não é mais o foco central. No entanto, tais técnicas não são postas em prática. A Figura 4.1 apresenta a ideia de arquitetura proposta no artigo para o controle de acesso granular:

Figura 4.1. Arquitetura de implementação proposta



Fonte: Adaptado de Agrawal et al, 2005.

De acordo com a Figura 4.1, o Tradutor das Políticas aceita a Política de Privacidade (escrita na linguagem P3P, por exemplo). Além disso, aceita metadados armazenados em catálogos de privacidade e gera restrições no nível de linha e/ou célula que implementam a política. As tabelas PT e PR armazenam informações de nome de tabelas, colunas e informações sobre os tipos P3P que foram o Catálogo de Privacidade e são utilizados como metadados de entrada para o Tradutor das Políticas.

4.2 Autorização granular através de Grants com predicados

Surajit et al. (2007) propõem um modelo no qual o controle de acesso granular aos dados é baseado na adição de predicados aos comandos de controle dos dados GRANT. Na linguagem de Banco de Dados SQL (Melton, 1994) os comandos GRANT e REVOKE são responsáveis por permitir (GRANT) e revogar (REVOKE) privilégios dos usuários. No entanto não existe, atualmente, a possibilidade de adicionar predicados a tais comandos de forma que possibilite a implementação do controle de acesso granular. Um comando GRANT para dar privilégio de leitura para todos os usuários em uma tabela que armazena dados de funcionários, por exemplo, pode ser visualizado abaixo:

Código 4.7. Exemplo de comando GRANT da Linguagem DCL.

```
GRANT SELECT ON EMPLOYEE TO PUBLIC;
```

Na proposta do artigo, o comando acima poderia ser estendido para prover acesso granular aos dados. Por exemplo, caso fosse necessário dar o mesmo privilégio acima, mas considerando que cada usuário só tivesse acesso aos seus dados, e não a todos os dados da tabela, a permissão de acesso poderia ser criada a partir do comando abaixo:

Código 4.8. Exemplo de comando GRANT para acesso granular de linhas na proposta de Surajit et al. (2007).

```
GRANT SELECT ON EMPLOYEE  WHERE (EMPID = USERID())
TO PUBLIC;
```

Além disso, é possível permitir o acesso a determinadas colunas e linhas ao mesmo tempo, conforme exemplo abaixo:

Código 4.9. Exemplo de comando GRANT para acesso granular de linhas e colunas na proposta de Surajit et al. (2007).

```
GRANT SELECT ON EMPLOYEE(NAME)
WHERE (DEPT ='SALES') TO PUBLIC
```

A partir do método proposto, também é possível criar grupos de usuários para facilitar a administração das políticas de segurança em grupos hierárquicos, conforme exemplo abaixo:

Código 4.10. Exemplo de criação de grupos de usuários na proposta de Surajit et al. (2007).

```
CREATE GROUP EMPLOYEEGRP AS (…)
CREATE GROUP MANAGERGRP AS EMPLOYEEGRP UNION (…)
```

Após a criação dos grupos de usuários, pelo método proposto no artigo, é necessário criar grupos de autorização para definir um conjunto de autorizações em objetos relacionados, onde cada grupo deve possuir um relacionamento denominado *root* com o objeto (tabela) principal do grupo. O objetivo é evitar relacionamentos cíclicos. Abaixo é listado um exemplo de um grupo de autorização descrito no artigo, onde o objeto principal (*root*) é a tabela *order*, que se relaciona com mais quatro tabelas (*lineitem*, *part*, *partsupp*, *supplier*):

Código 4.11. Exemplo de criação de grupos de autorização na proposta de Surajit et al. (2007).

```
create authorization select_purchaseorder
with root order O as (
select on order O,
select on lineitem L where
(L.order_id=O.order_id)),
select on part P where
(P.part_id=L.part_id),
select on partsupp PS where
(PS.part_id = P.part_id),
select on supplier S where
(S.supplier_id = PS.supplier_id))
```

Para finalizar, é necessário permitir o acesso aos grupos de autorização para os grupos de usuários, conforme exemplo abaixo:

Código 4.12. Exemplo de comando para associar grupos de autorização aos grupos de usuários na proposta de Surajit et al. (2007).

```
grant select_purchaseorder
where(purchaser_id = userId())
to employeeGrp
grant sel_update_purchaseorder
where (purchaser_id in
(select user_id from employee, manager
where employee.deptid=manager.deptid
and manager.mgrid = userId()))
to managerGrp
```

A proposta teórica é simples, interessante e possui as seguintes características, destacadas como vantagens desta abordagem:

- Semântica clara e simples;
- Compatibilidade com o mecanismo de segurança atualmente utilizado pelos principais fornecedores de SGBD do mercado;
- Fácil administração;
- Permite autorização ao nível de linha e de coluna; e
- Não necessita criar visões. Por este motivo, possui baixo impacto no código da aplicação.

No entanto, a ideia dos autores, como próximo passo, é discutir e negociar com os fornecedores de sistemas de bancos de dados a implementação da proposta descrita no artigo como extensão aos mecanismos de controle de acesso atuais de cada SGBD, ou seja, mesmo que a proposta seja aprovada, ficar-se-ia, mais uma vez, dependente dos contratos possivelmente abusivos de licenças de *software*.

4.3 Bouncer: Controle de Acesso Granular baseado em Políticas para grandes bancos de dados.

Opyrchal et al (2011) desenvolveram um mecanismo prático chamado *Bouncer* para realizar o controle de acesso granular aos dados. A solução proposta utiliza uma abordagem introduzida por Blaze et al. (1996) chamada *Trust Management* (gerência da

confiança) que consiste em uma linguagem para a definição e avaliação de políticas de controle de acesso.

Existem alguns sistemas que utilizam o modelo de gerência de confiança. O Bouncer utiliza o CPOL, onde as políticas são escritas em classes da linguagem de programação C++. Através dele, as regras são escritas em uma interface Web e, posteriormente, transcritas para classes C++. Os campos necessários para se definir uma política através do CPOL são listados no Quadro 4.1.

Quadro 4.1. Campos necessários para definição de uma política usando o CPOL – Adaptado de (Opyrchal et al, 2011).

CPOL - Campos de acesso às regras	
Campo	Descrição
Proprietário (*Owner*)	O proprietário (O*wner*) é a entidade cujos recursos são controlados pela regra.
Licenciado(s)	O licenciado é uma entidade ou grupo que receberá os privilégios de acesso. Caso múltiplos licenciados sejam especificados, então todos os licenciados devem requisitar o acesso juntos para que a regra seja aplicada.
Ficha de Acesso (*AcessToken*)	A ficha de acesso (*AccessToken*) contém informações sobre os privilégios associados a esta regra.
Condição	O CPOL verifica se a condição é verdadeira (*True*) antes de permitir o acesso (*AcessToken*) ao objeto alvo.

A partir do conceito dos campos citados acima é apresentado no artigo um exemplo de como criar uma política de acesso cujo objetivo é fazer com que Bob só tenha acesso aos dados de Alice no nível de prédio das 9h às 17h de segunda a sexta-feira, ou seja, Bob saberá se Alice está no prédio (mas não em que quarto) somente nos dias e horários pré-determinados. Para este exemplo, a política mostrada na Figura 4.2 poderia ser criada usando o CPOL:

Figura 4.2. Exemplo de criação de política usando o CPOL

```
Owner: Alice
Licensee: Bob
AccessToken{
    LocationResolution = Building
    IdentityReslution = Name
    DelegationPrivileges = None }
Condition {
    AfterTime = 9AM
    BeforeTime = 5PM
    Days = {Monday, Tuesday, Wednesday, Thursday,
            Friday}
    Building = Benton Hall }
```

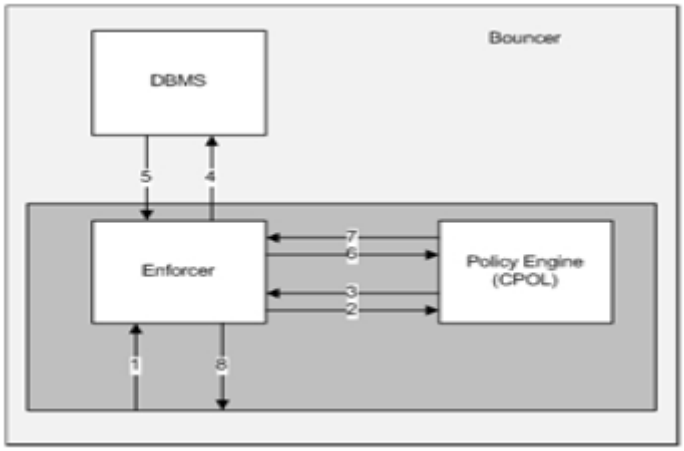
Fonte: Opyrchal et al, 2011.

Além do componente CPOL que é responsável pelo cadastro das políticas de acesso, a aplicação desenvolvida possui um módulo chamado *Enforcer* que realiza a interpretação e determina o cumprimento das permissões descritas nos *AccessToken* definidas nas políticas. O mecanismo de aplicação da política de segurança descrito no artigo segue o seguinte fluxo:

1. Um comando SQL é enviado pela aplicação e interceptado pelo *Enforcer*;
2. *Enforcer* verifica, através do CPOL, se o comando é permitido;
3. Se o comando não for permitido, um erro é retornado para a aplicação. Senão, a consulta segue para ser executada pelo sistema gerenciador de banco de dados;
4. Caso permitido, a consulta segue para o SGBD;
5. O SGBD executa o comando SQL e retorna o resultado para o módulo *Enforcer*;
6. O módulo *Enforcer* verifica a política de segurança associada ao resultado do SGBD através do módulo *Policy Engine*;
7. O módulo *Policy Engine* retorna a avaliação do resultado. Dependendo desta avaliação, o módulo *Enforcer* pode apagar registros, modificar campos ou permanecer com o registro original; e
8. Após o ajuste ou não dos registros, eles são retornados para a aplicação.

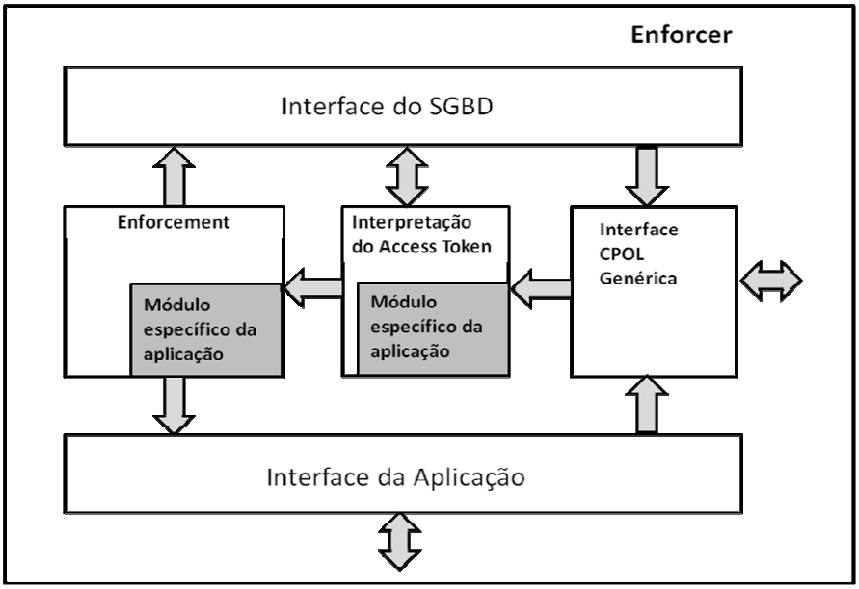
Os componentes (módulos) e seus respectivos relacionamentos descritos no fluxo acima podem ser visualizados na Figura 4.3. Já a Figura 4.4 apresenta o detalhamento do módulo *Enforcer* descrito.

Figura 4.3. Macro Componentes do Bouncer



Fonte: Opyrchal et al, 2011.

Figura 4.4. Detalhamento do Componente Enforcer.



Fonte: Adaptado de Opyrchal et al, 2011.

O diferencial desta solução para as demais é o seu aspecto prático, ou seja, foi desenvolvido um protótipo chamado Bouncer Database System (Opyrchal et al, 2011) implementado utilizando os sistemas gerenciadores de banco de dados MySQL

(MYSQL, 2012) e PostgreSQL (PostgreSQL, 2012), o componente *Enforcer* na linguagem C++ e o CPOL como sistema de avaliação das políticas. Além disso, existe uma interface Web para definição das políticas. No entanto, a solução proposta implementa o controle de acesso granular, armazena os dados retornados pelo comando SQL em memória e remove os dados inacessíveis antes de devolver o resultado, causando, desta forma, perda de desempenho.

4.4 Comparativo entre as metodologias

O Quadro 4.2 traz um comparativo entre a metodologia proposta, denominada Grão, e as metodologias abordadas neste capítulo. Os campos exibidos no quadro e as características comparadas são as seguintes:

- Independência do SGDB – Indica se o mecanismo de controle é independente do SGBD utilizado. No caso onde o mecanismo proposto resida dentro do gerenciador de banco de dados, será julgada como dependente;
- Implementação prática – Indica se existe implementação (protótipo) disponível para avaliação;
- Controle de acesso por coluna – Indica se o mecanismo de controle criptografa ou suprime colunas com conteúdo protegido;
- Controle de acesso por linha - Indica se o mecanismo de controle suprime linhas com conteúdo protegido;
- Ferramenta para gerenciamento das políticas - Indica se existe alguma ferramenta para gerenciamento e controle das políticas de acesso aos dados;
- Alteração do modelo de dados – Indica se a solução necessita que o modelo de dados atual seja alterado adicionando colunas que servirão como indicador do nível de segurança de uma linha;
- Múltiplas políticas por tabela – Indica se o mecanismo de controle de acesso permite mais de uma política por objeto (tabela) ao mesmo tempo, adicionando predicados;
- Retorna erro para aplicação caso o comando SQL não seja permitido – Indica se o mecanismo proposto retorna algum tipo de erro para a aplicação em caso de tentativa de acesso a um dado não autorizado. Em

caso negativo, a abordagem indica que o usuário não terá acesso aos dados, mas não saberá, explicitamente, que tentou acessar informações não autorizadas; e

- Perda de desempenho – Indica se o mecanismo proposto causa perda de desempenho na devolução dos dados ao usuário final.

Quadro 4.2: Quadro comparativo entre a metodologia proposta, as metodologias descritas e as características disponíveis em relação ao controle de acesso granular.

Metodologia / Característica	Predicate Grants	Extending DBMS	Bouncer	Grão
Independência do SGDB	Não	Não	Sim	Não *
Implementação prática	Não	Não	Sim	Sim
Controle de acesso por coluna	Sim	Sim	Sim	Sim
Controle de acesso por linha	Sim	Sim	Sim	Sim
Ferramenta para gerenciamento das políticas	Não	Não	Sim	Sim
Alteração do modelo de dados	Não	Não	Não	Não
Múltiplas políticas por tabela	Sim	Sim	Sim	Sim
Retorna erro para aplicação caso o comando SQL não seja permitido	Não	Não	Sim	Não
Perda de Desempenho	Não	Não	Sim	Não

* É, entretanto, independente do SGBD open source utilizado.

4.5 Considerações Finais

Trabalhos recentes ainda são realizados sobre o tema do controle de acesso granular. Entre eles, De Capitani di Vimercati et al (2008), propuseram uma solução baseada em permissões e visões gráficas. O mecanismo verifica se o comando SQL pode ser executado validando o comando a partir de um conjunto de permissões válidas.

No mesmo ano, Olson et al (2008) propôs um novo modelo de controle de acesso chamado Reflective Database Access Control (RDBAC), no qual os autores detalham um framework para criação e controle das políticas de acesso.

A existência de estudos recentes indica que nenhuma solução foi conceitualmente aceita como padrão pela comunidade científica ou utilizada em larga escala pelos principais fornecedores de sistemas de banco de dados *open source* existentes no mercado. Além disso, nenhuma das soluções descritas possibilita a utilização do mecanismo de controle de acesso granular para banco de dados embarcados *open source*. Desta forma, a partir do próximo capítulo, será detalhada uma nova metodologia de acesso granular aos dados voltada para sistemas de banco de dados relacionais *open source.*

Capítulo 5: Grão – Uma Metodologia para controle de Acesso Granular a SGBD Relacionais *Open Source*

Neste capítulo será descrita a metodologia Grão proposta por este trabalho. Desta forma, serão apresentados os objetivos e a conceituação básica da referida metodologia. Além disso, serão detalhadas as principais etapas da metodologia proposta tais como o Controle de Conexão ao SGBD, Interceptação do Comando SQL, Análise Granular do Comando SQL e Reescrita do Comando SQL. Também será apresentado um exemplo de utilização da metodologia bem como os algoritmos propostos.

Por fim, serão discutidos os aspectos positivos e as limitações da referida solução e apresentada a conclusão do referido capítulo.

5.1 Objetivos da Metodologia

Várias abordagens têm sido sugeridas para realizar o controle de acesso granular aos dados em um sistema de banco de dados relacional. Algumas delas são características dos atuais líderes do mercado de SGBD que possuem um alto custo de licenciamento conforme listado no Quadro 5.1. Outras são novos conceitos que poderiam ser introduzidos no modelo de autorização do controle de acesso dos fornecedores de SGBD caindo no mesmo problema de licenciamento, e a minoria tem adotado técnicas práticas para interceptar e reescrever o comando SQL a partir de uma política de segurança previamente definida (Opyrchal et al, 2011).

Quadro 5.1: Preço (por processador) aproximado da versão Enterprise para os bancos de dados Oracle, DB2 e SQL Server baseada em (SQL Server White Paper, 2010).

	SQL Server	Oracle	IBM DB2
Licenciamento da Versão Enterprise (Incluso um ano de suporte)	$34,369	$47,500	$40,500
Opção para Ferramentas de Gerenciamento	Incluso	$5,000 (Diagnostics Pack) $5,000 (Tuning Pack) $5,000 (Configuration Mgmt. Pack) $3,500 (Change Mgmt. Pack) $3,500 (Patch Automation Pack) Total = $22,000	$15,300 (Performance Optimization Feature)
Opção de Segurança	Incluso	$11,500 (Advanced Security Option)	$11,100 (Advanced Access Control Feature)
Opção de Compressão	Incluso	$11,500 (Advanced Compression Option)	$15,300 (Storage Optimization Feature)
Opção de Dados Espaciais	Incluso	$17,500 (Spatial Option)	$11,100 (Geodetic Data Management Feature)
Opção de Replicação	Incluso	Incluso	$11,100 (Homogeneous Replication Feature)
Custo Total (em dólares)	**$34,369**	**$110,000**	**$104,400**

A metodologia Grão possui como objetivo principal permitir o controle de acesso granular aos dados, independentemente do SGBD relacional *open source* utilizado. Ela é composta de fluxos, controles, guias e responsáveis buscando proteger o acesso indevido aos dados em um banco de dados. Além disso, a metodologia define uma série de atividades necessárias para se criar um conjunto de políticas de segurança, cujo objetivo é traduzir as regras de acesso aos dados de uma organização em metadados que serão utilizados como guia para se definir que porção dos dados um determinado usuário poderá acessar em um SGBD.

A metodologia proposta também visa eliminar a necessidade de adquirir licença de software para obter um mecanismo de controle de acesso granular aos dados. A metodologia Grão foi baseada nos métodos descritos por Opyrchal et al (2011) e por (Chang e Hill, 2012), cujo título é "Método e aparato para reescrita de consultas com atributos auxiliares em operações de processamento de consultas". O método foi patenteado em fevereiro de 2012.

Grão também ajuda a gerenciar e facilitar a visibilidade das regras de acesso aos dados de uma organização, definindo procedimentos, regras e responsáveis pelo controle de acesso aos dados.

5.2 Arquitetura da Metodologia Grão

Para controlar o acesso granular aos dados, buscou-se reunir as melhores características fornecidas pelos atuais fornecedores de SGBD do mercado (Rayns et al , 2005; Sybase Technical Library, 2005; Oracle Corporation, 2011), bem como as melhores soluções e ideias disponibilizadas atualmente na literatura (Agrawal et al.,2005; Surajit et al.,2007; Opyrchal et al.,2011) para os problemas do controle de acesso, em particular o controle de acesso granular aos dados. No entanto, conforme descrito na seção anterior, a metodologia Grão foi baseada nos métodos descritos por Opyrchal et al (2011) e por (Chang e Hill, 2012).

Na metodologia proposta, procurou-se resolver problemas e preencher as lacunas que existem atualmente na literatura e no mercado de sistemas de banco de dados, de forma que o modelo permita o controle de acesso granular único a SGBD distintos, desde que o SGBD seja *open source*. Para atingir este objetivo, considerou-se importante descrever alguns conceitos utilizados na abordagem proposta, tais como:

- Esquema (*Schema*) – Conceito do usuário ou banco de dados proprietário da informação. Como podem existir várias tabelas de mesmo nome em uma mesma instância do SGBD, tal conceito é essencial para identificar e direcionar o acesso ao objeto correto;
- Comando SQL – Texto contendo o comando SQL utilizado para consultar (*SELECT*) ou realizar operações de manipulação de dados (*INSERT*, *UPDATE* e *DELETE*) em uma tabela ou visão no banco de dados;
- Objeto – Conceito que identifica tabelas e visões como objetos válidos ao controle de acesso;
- Objeto Principal – Como o próprio nome sugere, este conceito identifica o objeto principal ao qual a política está relacionada para limitar (ou não) o acesso aos dados e colunas;
- Objeto *Lookup* – Este conceito permite que os objetos principais se relacionem (geralmente pela chave estrangeira) com vários objetos *lookup* que devem auxiliar a definição do predicado que será criado e adicionado ao comando SQL;

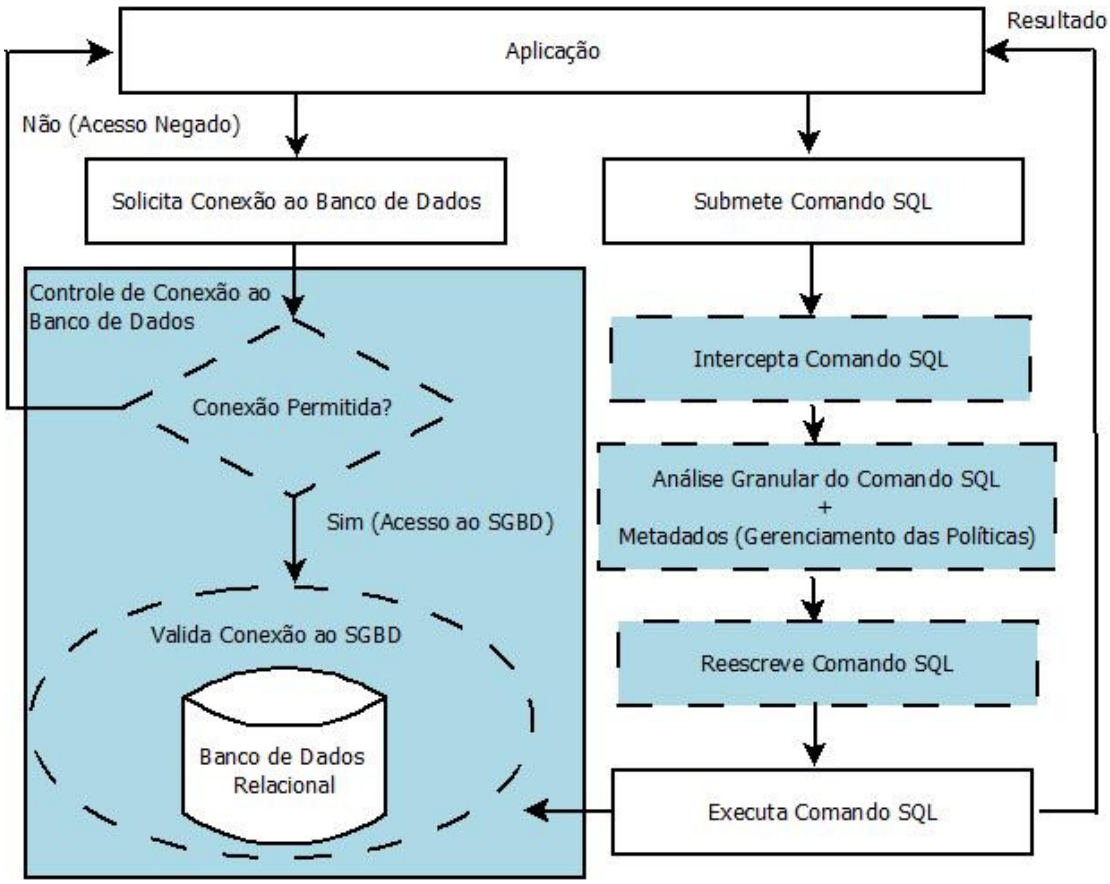
- Usuário – Conceito do usuário do banco de dados que está tentando acessar um determinado Objeto em um determinado *Schema*;
- Filtro – Conceito que permite incluir manualmente uma restrição à política que está sendo criada. As restrições podem utilizar operadores lógicos e de comparação;
- Predicado – Conceito que considera a montagem da cláusula *WHERE* obtendo a transformação dos filtros e relacionamentos oriundos dos Objetos *Lookup*;
- Coluna Invisível – Conceito que permite esconder informações (atributos) específicas dos Objetos Principais;
- Perfil de Acesso – Conceito utilizado para agrupar usuários com as mesmas características no acesso aos dados;
- Política de Segurança – Conceito que nomeia a política de segurança que está sendo criada e informa se a mesma está ativa ou não. Além disso, devem-se associar as políticas de segurança aos objetos do banco de dados e aos respectivos perfis de acesso.

A estrutura da metodologia Grão, visualizada na Figura 5.1, é baseada nos conceitos descritos anteriormente e, principalmente, nas etapas criadas para esta metodologia, descritas abaixo:

- Interceptação do Comando SQL – Etapa do mecanismo que interrompe o curso natural do Comando SQL;
- Controle de Conexão ao Banco de Dados – Consiste no método de controle da conexão ao banco de dados. Geralmente está associado ao controle de um programa de computador que faz a ligação entre a aplicação (cliente) e o banco de dados (servidor);
- Análise Granular do Comando SQL – Método que permite verificar se o Comando SQL necessita de ajustes, em termos de segurança, antes de ser executado no Banco de Dados. Os ajustes podem limitar o acesso aos dados no nível de linha e/ou de coluna;
- Reescrita do Comando SQL – Representa o método que permite alterar (ou não) o comando SQL enviado originalmente; e
- Gerenciamento das Políticas de Segurança – Etapa da metodologia que tem como objetivo representar as regras (políticas) de segurança de uma

organização em metadados que serão utilizados pela etapa de Análise Granular do Comando SQL.

Figura 5.1. Arquitetura da Metodologia Grão.



Fonte: Elaborado pelo autor.

Na Figura 5.1, as formas tracejadas indicam controles realizados pela metodologia proposta. As demais formas fazem parte do mecanismo tradicional de conexão a um SGBD relacional por um aplicativo.

5.3 Detalhamento das Etapas da Metodologia

Conforme descrito na Seção 5.2, existem cinco fases ou etapas da Metodologia Grão (Interceptação do Comando SQL, Análise Granular do Comando SQL, Reescrita do Comando SQL, Controle de Conexão ao SGBD e Gerenciamento das Políticas de Segurança). Cada uma das fases possui objetivos bem definidos e grande parte delas

está inter-relacionada. Nas subseções 5.3.1 a 5.3.5 será detalhada cada uma das etapas da Metodologia Grão.

Assim, cada etapa é apresentada com as principais características, relacionamentos e objetivos. Ao final de cada etapa é apresentado um fluxograma de funcionamento da mesma. Em cada fluxograma, as formas tracejadas indicam controles realizados pela metodologia proposta. As demais formas fazem parte do mecanismo tradicional de conexão a um SGBD relacional por um aplicativo. Após a apresentação de todas as etapas da metodologia, será apresentado na subseção 5.3.6 um exemplo de utilização da mesma.

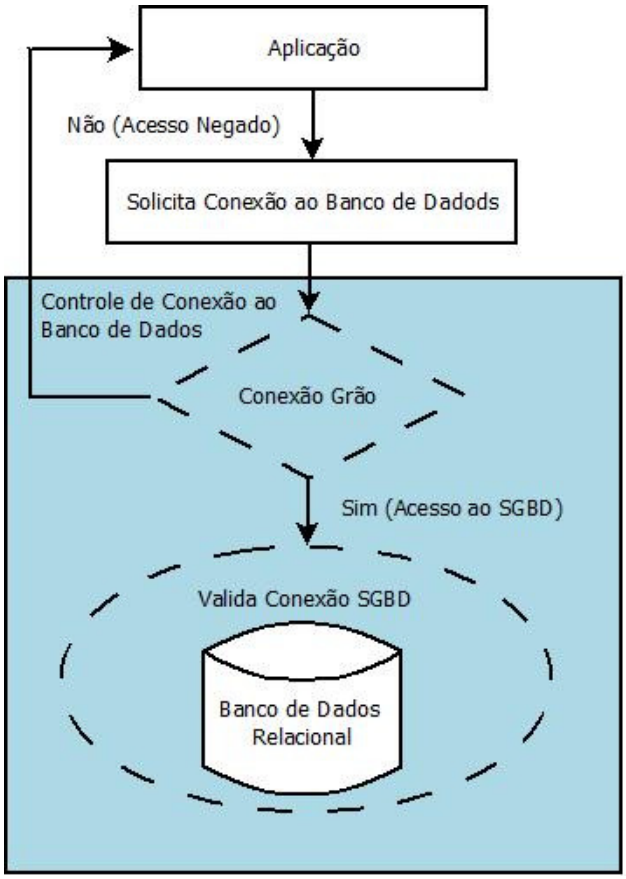
5.3.1. Controle de Conexão ao SGBD

A etapa de Controle de Conexão ao SGBD tem como objetivo principal permitir o acesso ao SGBD apenas para conexões "autorizadas". O processo de autorização consiste em dois mecanismos, um chamado de Conexão Grão e o outro chamado de Valida Conexão SGBD.

O mecanismo Conexão Grão, nesta etapa, tem as funcionalidades de prover o mecanismo de conexão ao SGBD Relacional e fornecer ao SGBD a informação que a conexão é oriunda de um mecanismo autorizado.

O mecanismo Valida Conexão SGBD deve garantir que as conexões ao SGBD utilizem o mecanismo de conexão chamado Conexão Grão, determinando em caso contrário que o SGBD rejeite as conexões oriundas de outros mecanismos. Como as conexões realizadas através do mecanismo autorizado são devidamente informadas ao SGBD, é possível identificar as conexões não autorizadas e, desta forma, não permitir o acesso. A Figura 5.2 exibe o fluxograma da etapa de Controle de Conexão ao SGBD:

Figura 5.2. Etapa do Controle de Conexão ao SGBD.



Fonte: Elaborado pelo autor.

A etapa de Controle de Conexão ao SGBD é necessária para identificar e validar o mecanismo de conexão utilizado. Sem esta etapa, o método proposto não pode garantir que as políticas de segurança serão impostas e, por consequência, não garante o controle de acesso granular aos dados.

5.3.2. Interceptação do Comando SQL

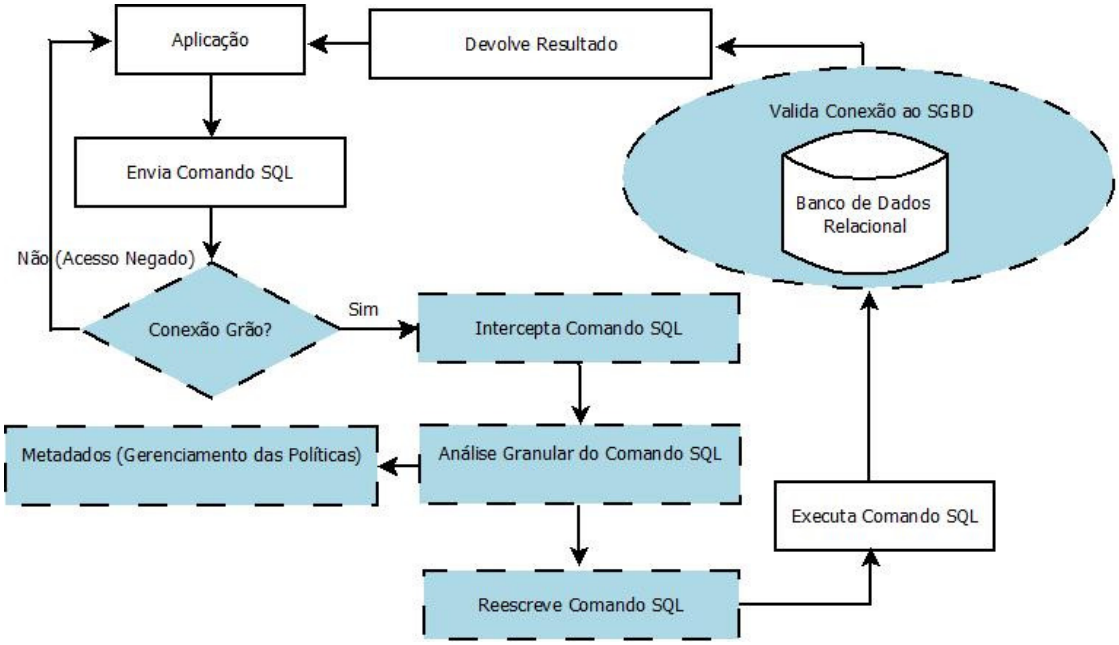
A etapa de Interceptação do Comando SQL tem como objetivo principal interromper o curso natural do Comando SQL para permitir realizar as alterações necessárias no referido comando, antes de sua execução no banco de dados.

A interceptação é realizada pelo mecanismo nomeado de Conexão Grão. Ela ocorre antes da execução do Comando SQL no banco de dados para permitir que, caso exista alguma Política de Segurança para o Usuário da conexão, tal comando seja reescrito para que possa considerar essa política.

É importante frisar que tal interceptação só ocorre em conexões que utilizem o mecanismo Conexão Grão. Todas as demais conexões não possuem a segurança granular ativa e, por este motivo, devem ser recusadas pelo SGBD.

Além de interceptar o Comando SQL, essa etapa se relaciona com a etapa de Análise Granular do Comando SQL, a qual será descrita na Seção 5.3.3. A Figura 5.3 detalha o fluxograma da etapa de Interceptação do Comando SQL:

Figura 5.3. Etapa da Interceptação do Comando SQL.



Fonte: Elaborado pelo autor.

A etapa de Interceptação do Comando SQL é essencial para as duas etapas seguintes (Análise Granular do Comando SQL e Reescrita do Comando SQL). Esta etapa, existente na maioria das metodologias propostas, torna-se imprescindível pela necessidade de reescrever o comando SQL após a interceptação de acordo com o nível de acesso do usuário em relação ao comando SQL originalmente enviado.

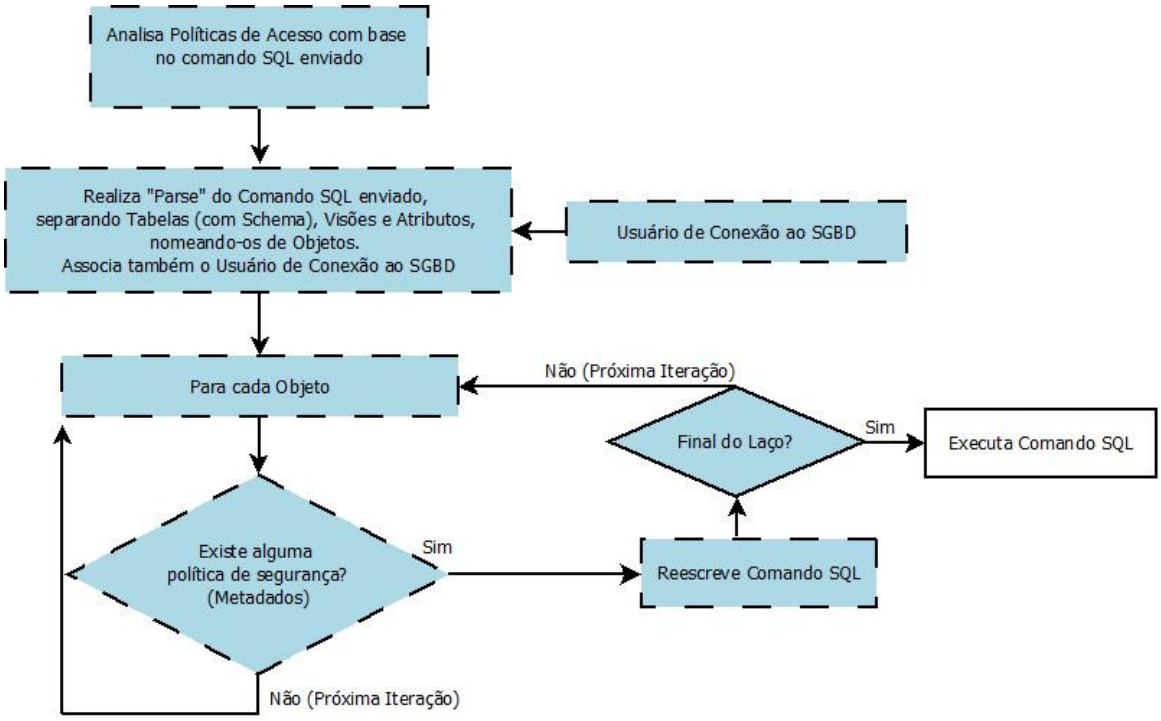
5.3.3. Análise Granular do Comando SQL

A etapa de Análise Granular do Comando SQL tem como objetivo principal se comunicar com as etapas de Gerenciamento das Políticas de Segurança e Reescrita do Comando SQL. Por este motivo, esta etapa tem a função de realizar o "*parse*" do

Comando SQL enviado, separando tabelas, visões e atributos e os nomeando de Objetos. Além disso, esta etapa recebe o usuário de conexão ao SGBD.

A análise granular, após coletadas todas as informações (objetos + usuário), é realizada por um laço que, para cada iteração, se comunica com a etapa de Gerenciamento das Políticas de Segurança, enviando as informações necessárias para receber como retorno o predicado (filtro do Comando SQL) que será adicionado pela etapa nomeada de Reescrita do Comando SQL. A Figura 5.4 detalha o fluxograma da etapa de Análise Granular do Comando SQL:

Figura 5.4. Etapa da Análise Granular do Comando SQL.



Fonte: Elaborado pelo autor.

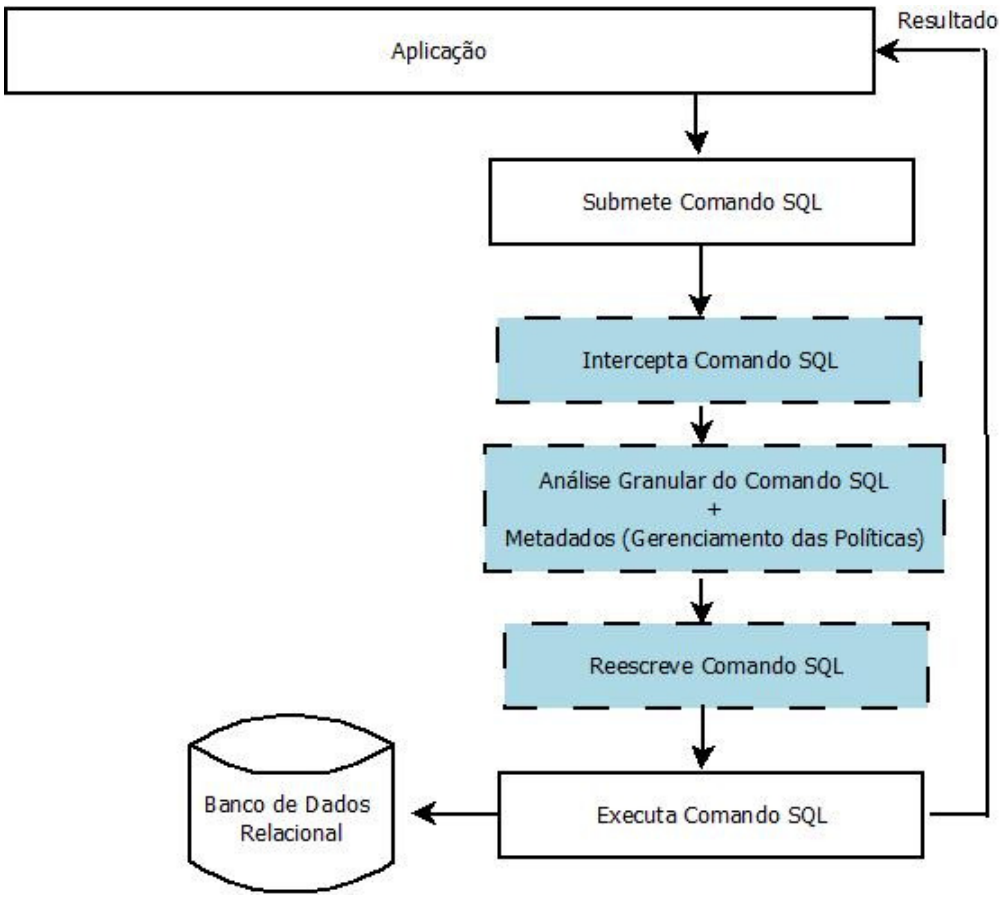
A Análise Granular do Comando SQL é a etapa com maior grau de complexidade e inteligência da metodologia proposta, pois é necessário traduzir as políticas de segurança (armazenadas em metadados) em predicados, filtros e alterações no Comando SQL original. Sem esta etapa, seria necessário criar as políticas de segurança a partir de comandos SQL pré-definidos, o que tornaria o método bem mais dependente do conhecimento da sintaxe SQL por parte da equipe responsável pela criação das políticas de segurança.

5.3.4. Reescrita do Comando SQL

A etapa de Reescrita do Comando SQL tem como objetivo principal receber o predicado gerado a partir da etapa de Análise Granular do Comando SQL e alterar o Comando SQL original, adicionando o predicado recebido (controle de acesso por linha) e ajustando os campos que não devem ser visíveis (controle de acesso por coluna).

A Reescrita do Comando SQL, após receber e adicionar o predicado (filtro) devolve o Comando SQL alterado para a fase de Execução do Comando SQL, a qual foi inicialmente desviada através da etapa de Interceptação do Comando SQL. A partir deste momento, o método devolve o controle da execução do Comando SQL ao processo natural do seu funcionamento conforme visualizado na Figura 5.5:

Figura 5.5. Etapa Reescrita do Comando SQL.



Fonte: Elaborado pelo autor.

A etapa de Reescrita do Comando SQL é a última etapa do processo, sendo necessária para consolidar as etapas anteriores no objetivo final: modificar dinamicamente o comando SQL.

5.3.5. Gerenciamento das Políticas de Segurança - Metadados

A etapa de Gerenciamento das Políticas de Segurança tem como principal objetivo fornecer os metadados necessários para a etapa de Análise Granular do Comando SQL, ou seja, esta etapa tem papel fundamental para que o processo de reescrita do Comando SQL esteja de acordo com as normas de segurança da Organização. Por este motivo, esta etapa deve ser realizada por uma pessoa ou equipe chamada de Administradores do Acesso Granular aos dados. O papel desta equipe é transformar as regras de segurança da organização em metadados.

Para realizar o processo de criação de uma política de segurança de acesso e fornecer os metadados necessários através do mecanismo proposto, a etapa do Gerenciamento das Políticas de Segurança foi dividida nos seguintes passos:

Definição dos Objetos e Atributos → Definição dos objetos (tabelas ou visões) e dos respectivos atributos que poderão participar das políticas de acesso. Para incluir um objeto, deve-se informar:

- Nome do *Schema* do Objeto;
- Nome do Objeto;
- Descrição do Objeto (Opcional);
- Tipo do Objeto (Tabela ou Visão); e
- Nome do atributo.

Definição dos Perfis de Acesso e dos Usuários Associados → Definição dos perfis de acesso com os respectivos usuários (de banco de dados ou aplicação) associados. Para incluir um perfil, deve-se informar:

- Código do Perfil;
- Nome do Perfil;
- Descrição do Perfil;
- Nome do Usuário; e
- Nome do Usuário Completo.

Definição da Política de Acesso → Neste momento é possível informar em que diferentes tipos de comandos DML (*INSERT*, *UPDATE*, *DELETE* e S*ELECT*) a política será executada e se a política está ativa ou não. Para incluir uma política, deve-se informar:

- Código da Política;
- Nome da Política;

- Descrição da Política;
- Dispara no *Select* (Sim/Não);
- Dispara no *Insert* (Sim/Não);
- Dispara no *Update* (Sim/Não);
- Dispara no *Delete* (Sim/Não); e
- A política está ativa (Sim/Não).

Associação dos Objetos às Políticas → Associar os objetos (tabelas ou visões) às políticas de segurança. Para incluir uma associação da política ao objeto, deve-se informar:

- Nome do *Schema* do Objeto;
- Nome do Objeto;
- Condição de Acesso (Só incide sobre o objeto principal. Tudo – Todas as linhas do Objeto; Nada – Nenhuma linha do Objeto; Via Atributos – Será necessário utilizar valores específicos para algum atributo do objeto para filtro);
- Função do Objeto (Objeto Base – Objeto cuja operação DML vai incidir; Objeto *Lookup* – Objeto opcional utilizado como algum filtro extra);
- *Schema* de Referência – Nome do *Schema* de Referência para objetos *Lookups* (Utilizado para unir os objetos bases e *lookups*); e
- Nome do Objeto de Referência - Nome do objeto de Referência para objetos *Lookups* (Utilizado para unir os objetos bases e *lookups*).

Definição dos Filtros → Criação dos filtros para os objetos associados à política. Além disso, determina se uma ou mais colunas serão visíveis no retorno do comando SQL *Select*. Para definir os filtros para objetos, deve-se informar:

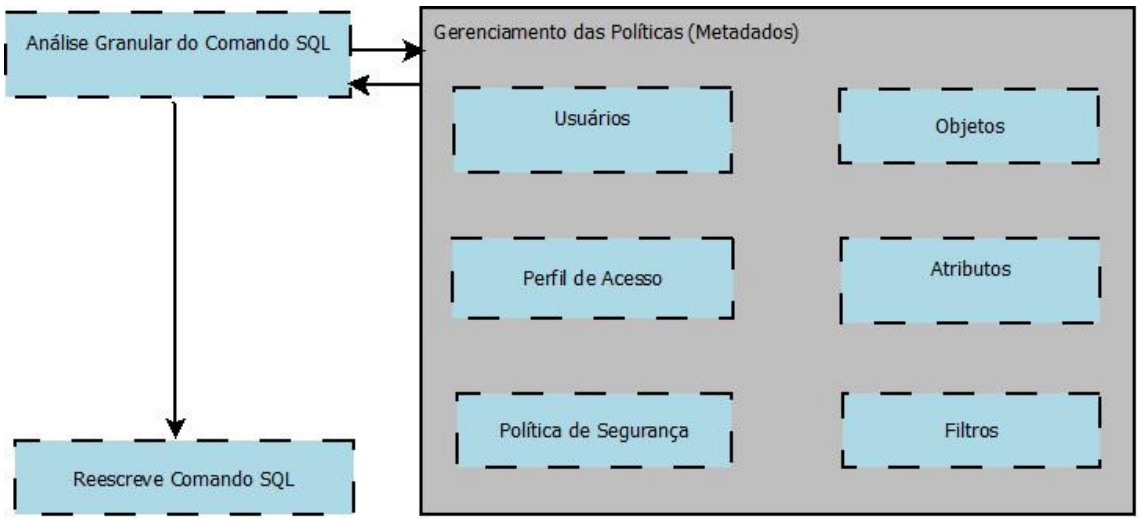
- Nome do Atributo;
- Operador de Comparação (Maior, Menor, Igual, Entre, por exemplo);
- Valor do Atributo – Valor que será comparado ao atributo com base no operador de comparação;
- Valor Limite – Valor Máximo só utilizado no caso do operador de comparação ser o "entre" (*BETWEEN*); e
- Operador Lógico – No caso do filtro ser por mais de uma coluna ao mesmo tempo, deve-se incluir o operador lógico (*AND*, *OR* ou *NOT*).

Associação das Políticas aos Perfis de Acesso → Associar as políticas aos perfis de acesso que englobam um ou mais usuários do banco de dados / aplicação onde a regra deverá atuar. Para incluir uma associação da política ao objeto, deve-se informar:

- Código da Política.
- Código do Perfil.

A Figura 5.6 detalha o fluxograma da etapa de Gerenciamento das Políticas (Metadados):

Figura 5.6. Gerenciamento das Políticas de Segurança - Metadados.



Fonte: Elaborado pelo autor.

A etapa do Gerenciamento das Políticas de Segurança é muito importante para a metodologia proposta, pois fornece meios e padrões para criação e manutenção das regras e políticas de segurança de uma organização em relação aos dados. Caso esta etapa não existisse, a etapa de Análise Granular dos Comandos SQL teria uma complexidade muito maior do que a existente para prover os filtros e predicados de cada comando SQL.

5.3.6. Exemplo de utilização da Metodologia Grão

Objetivando facilitar a compreensão e ilustrar as etapas descritas anteriormente, será apresentado um exemplo simples de como a metodologia proposta realiza o controle de acesso granular aos dados.

O exemplo visa a construção de uma política de segurança cujo objetivo é fazer com que o usuário chamado USER01, do perfil RH, só tenha acesso aos dados dos empregados cadastrados na tabela FUNCIONARIOS cujo salário seja inferior a R$ 5.000,00. O Código 5.1 exibe o Comando SQL original (enviado pelo usuário / aplicação) e o Código 5.2 exibe o Comando SQL alterado (devolvido) após a atuação das etapas da metodologia Grão:

Código 5.1. Comando SQL original enviado pelo usuário / aplicação.

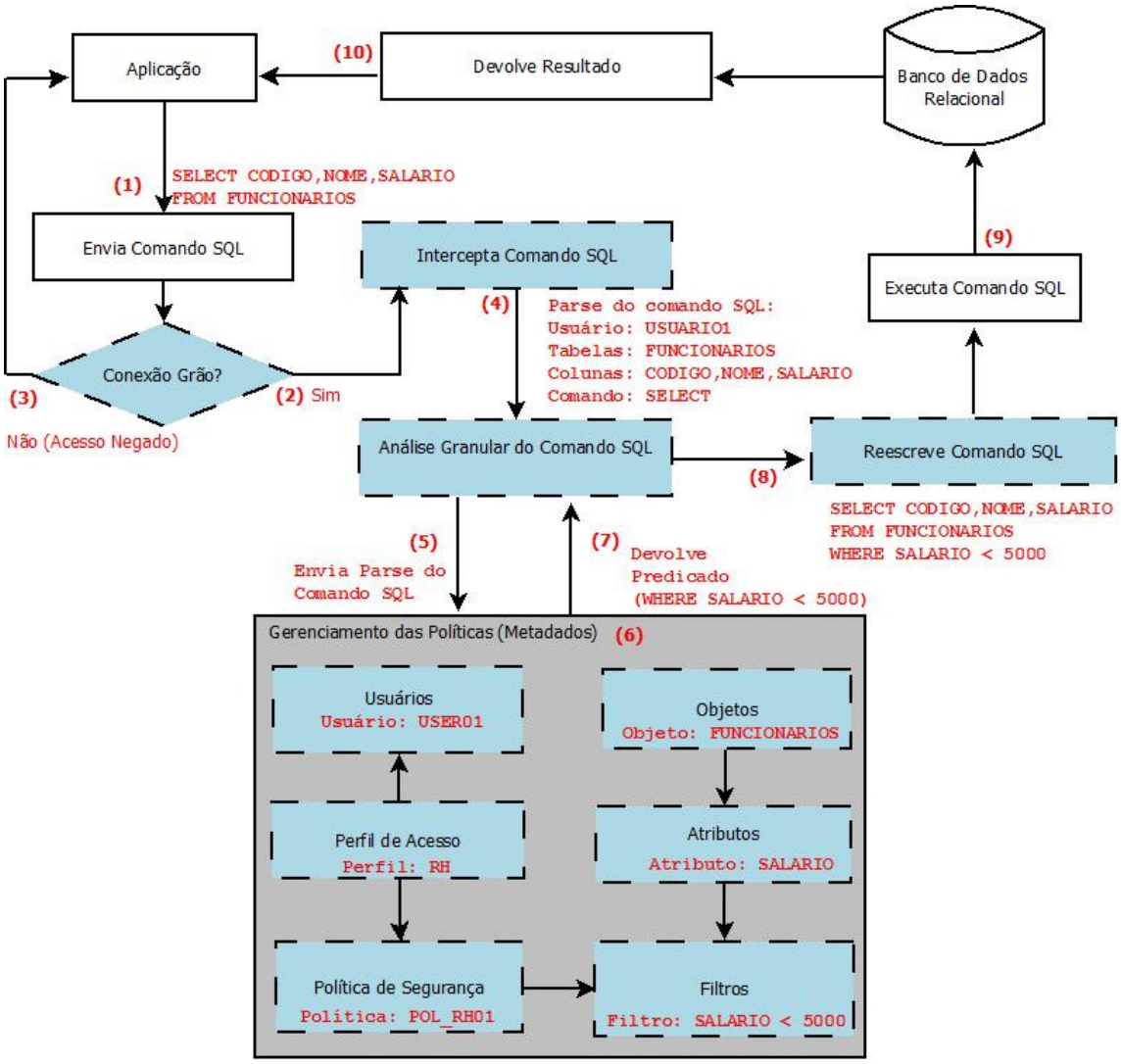
```
SELECT CODIGO,NOME,SALARIO FROM FUNCIONARIOS
```

Código 5.2. Comando SQL alterado (devolvido) após a atuação das etapas da metodologia Grão para o usuário RH.

```
SELECT CODIGO,NOME,SALARIO FROM FUNCIONARIOS
WHERE SALARIO < 5000
```

A partir do exemplo descrito, a Figura 5.7 descreve o funcionamento da metodologia Grão em cada uma de suas etapas. A sequência do fluxograma segue ordem crescente de um a dez, partindo do envio do comando SQL através da aplicação até a devolução do resultado do Comando SQL para a aplicação:

Figura 5.7. Exemplo de Utilização da Metodologia Grão.



Fonte: Elaborado pelo autor.

5.4 Algoritmos propostos para a solução

Para implementar as etapas descritas na metodologia, foram criados os algoritmos descritos nos códigos 5.3 a 5.7:

Código 5.3. Algoritmo da Solução Proposta – Etapa de Controle de Conexão ao SGBD-Conexão Cliente.

```
1: // Entrada: U, Usuário de Banco de Dados ou Aplicação
2: // Entrada: M, Modo de Conexão ao Banco de Dados (Conexão Grão ou
Demais Conexões)
3: // Entrada: I, ID de Conexão ao Banco de Dados
4: Se U PossuiControleGranular, então
5:    Se M é ConexãoGrão, então
6:     InformaBDConexaoPermitida(I,U)
7:    Fim Se
8: Fim Se
```

Código 5.4. Algoritmo da Solução Proposta – Etapa de Controle de Conexão ao SGBD-Conexão SGBD.

```
1: // Entrada: I, ID de Conexão ao Banco de Dados
2: // Entrada: M, Modo de Conexão ao Banco de Dados (Conexão Grão ou
Demais Conexões)
3: // Entrada: T, Tempo de intervalo entre as varreduras
4: A cada T tempo, execute
5:    Para cada I no BancoDeDados faça
6:          ValidaConexao(M) //Retorno X
7:          Se X é false, então FinalizaConexao(I)
8: Fim
```

Código 5.5. Algoritmo da Solução Proposta – Etapas de Interceptação do Comando SQL e Reescrita do Comando SQL.

```
1: // Entrada: S, Schema do Usuário
2: // Entrada: U, Usuário de Banco de Dados ou Aplicação
4: // Entrada: Q, Comando SQL
5: // Entrada: M, Modo de Conexão ao Banco de Dados (Conexão Grão ou
Demais Conexões)
6: Se M é ConexãoGrão, então
7:    AnaliseGranularComandoSQL(S,U,Q) //Retorno P (Predicado)
8:    Se P existe, então ReescreveComandoSQL(Q,P) //Retorno Q'
9: Fim
```

Código 5.6. Algoritmo da Solução Proposta – Etapa de Reescrita do Comando SQL.

```
1: // Entrada: Q, Comando SQL
2: // Entrada: P, Predicado (Filtro do Comando SQL)
3: Q' recebe Q Concatenado a P
4: Retorna Q'
9: Fim
```

Código 5.7. Algoritmo da Solução Proposta – Etapa de Análise Granular do Comando SQL.

```
1: // Entrada: S, Schema do Usuário
2: // Entrada: U, Usuário de Banco de Dados ou Aplicação
3: // Entrada: Q, Comando SQL
4: InicializarPerfil(S,U)
5: InicializarListaObjetos(Q) //Retorno OP (Objeto Principal), OL
(Objeto Lookup – Relacionamento)
6: Para cada ObjetoPrincipal faça
7:    InicializarListaPoliticasAtivasPrincipais(U,OP) //Retorno P
8:    BuscaPredicadoObjetoPrincipal(S,U,OP,P) //Retorno X
9:    BuscaInvisibilidadeColunaObjetoPrincipal(S,U,OP,P) //Retorno Y
10:   Se X existe, então AdicionaPredicadoPrincipal(Q,X)
11:   Se Y existe, então TornaInvisivelColunaObjetoPrincipal(Q,X)
12:   Para cada Objeto Lookup faça
13:         MontaJoin(S,U,OP,OL)
14:         BuscaPredicadoObjetoLookup(S,U,OL,P) //Retorno Z
15: Fim
17: retorne Predicado (P)
```

5.5 Considerações Finais

Foi descrita neste capítulo a metodologia proposta como solução para o controle de acesso granular em sistemas de bancos de dados relacionais *open source*. Além disso, foram detalhadas as principais etapas necessárias para atingir os objetivos da metodologia, bem como os algoritmos necessários para implementar tais etapas.

No capítulo seguinte é descrito o desenvolvimento da ferramenta Grão, que serve como prova de conceito para avaliar a metodologia criada, por meio de um experimento realizado com este fim.

Capítulo 6: Ferramenta Grão

Neste capítulo serão descritos os aspectos técnicos relacionados à construção e ao funcionamento da ferramenta Grão. Serão apresentados os modelos, arquitetura, tecnologias adotadas e analisadas as vantagens e limitações da mesma. Além disso, será realizado um breve descritivo sobre a ferramenta e um exemplo prático de utilização.

6.1 Descrição Geral da Ferramenta

Para validar a metodologia criada, optou-se por construir um protótipo de uma ferramenta que tem como objetivo principal controlar o acesso granular aos dados em bancos de dados relacionais *open source* a partir da intervenção do comando SQL enviado através do *driver* JDBC. As políticas de acesso aos dados estarão armazenadas em um banco de dados embarcado (H2) no próprio servidor de aplicação. Os conceitos de *driver* JDBC e de banco de dados embarcados serão expostos na seção 6.4.

Os principais diferenciais trazidos pela ferramenta Grão são:

- Independência do SGDB *open source* – Conforme descrito anteriormente, a grande maioria dos mecanismos de controle de acesso granular é específica para um determinado SGBD. O mecanismo criado pode ser facilmente adaptado para trabalhar com diversos SGBD que sejam compatíveis com o *driver* JDBC *open source*, sendo necessário apenas editar o *driver* JDBC e tratar particularidades de sintaxe SQL do SGBD escolhido;
- Controle de acesso por linha e coluna – A abordagem proposta permite o controle de acesso tanto no nível de linha quanto no nível de coluna;
- Ferramenta gráfica para gerenciamento das políticas – Foi criada uma interface gráfica que permite o cadastro, gerenciamento e controle das políticas de segurança aos dados;
- Alteração do modelo de dados – O mecanismo proposto não necessita a alteração do modelo de dados para controlar o acesso granular aos dados. No entanto, em algumas situações, será mais fácil e com melhor desempenho criar políticas de segurança após a alteração do modelo de dados; e

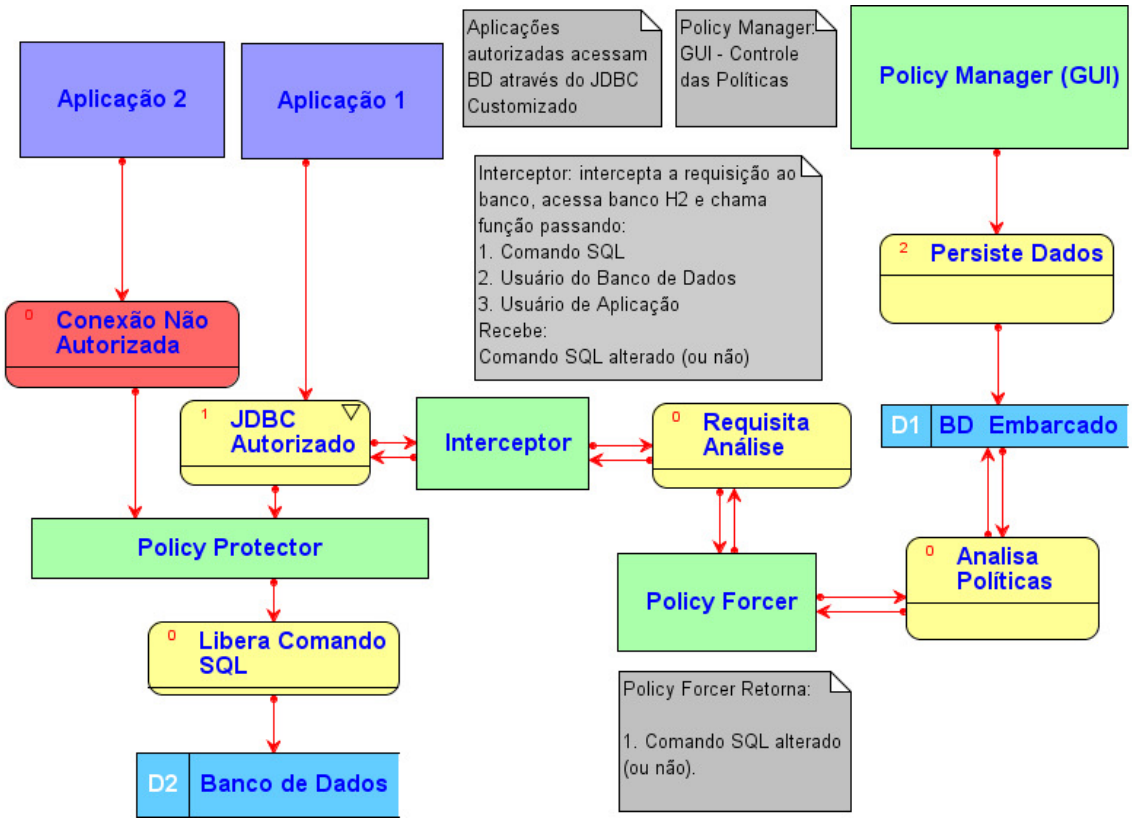
- Múltiplas políticas por tabela – A metodologia proposta permite a criação de mais de uma política por objeto ao mesmo tempo, adicionando N predicados ao comando SQL original.

6.2 Arquitetura

Conforme definido por Shaw e Garlan (2006), arquitetura de software fornece a descrição mais abstrata do software, envolvendo sua estrutura, comportamento e propriedades chave. Em um processo de desenvolvimento de software, a arquitetura do software desempenha um papel importante como ponte entre os requisitos e a implementação (Moraes e Vasconcelos, 2004).

A arquitetura geral da ferramenta Grão bem como suas relações com os demais sistemas (aplicações, banco de dados, por exemplo) são visualizados na Figura 6.1.

Figura 6.1 Arquitetura da Ferramenta Grão.



Fonte: Elaborado pelo autor.

A ferramenta Grão é composta de quatro (4) módulos principais. Dois deles estão embutidos no *driver* JDBC, um diz respeito à interface gráfica para gerenciamento

das políticas de segurança e o módulo restante (Policy Protector) não foi implementado. Abaixo são listados tais componentes e detalhado o funcionamento de cada um deles:

- Interceptor – Módulo responsável por interceptar o comando SQL na camada do *driver* de conexão com o banco de dados (*driver* JDBC). As interfaces Statement e PreparedStatement do *driver* JDBC foram estendidas com o objetivo de enviar a requisição de análise das políticas de seguranças para o componente Policy Forcer;
- Policy Forcer – Módulo que tem como função principal receber as informações necessárias (comando SQL, usuário, *Schema*) para verificar a análise granular do comando SQL, ou seja, se alguma política de segurança deve ser imposta. Em caso afirmativo, deve formatar o comando SQL alterado pela(s) política(s) de segurança e retornar o comando SQL alterado para o Interceptor;
- Policy Manager – Módulo responsável pelo controle, administração e gerenciamento das políticas de segurança e perfis de acesso. Possui uma interface Web na qual podem ser definidos todos os critérios de acesso granular aos dados. Os dados fornecidos a este módulo pelos usuários finais são persistidos em um banco de dados embarcado (H2); e
- Policy Protector – Módulo responsável por proibir o acesso ao banco de dados a partir de programas não autorizados, ou seja, aplicativos que tentem acessar o banco de dados sem utilizar o *driver* JDBC customizado devem ter acesso negado. Este módulo não foi implementado. No entanto, existem várias maneiras de proteger o acesso ao banco de dados de forma que o método seja respeitado. Abaixo são listadas duas maneiras de conseguir este controle:
    - Limitar o acesso ao banco de dados apenas pelo servidor de aplicação que utiliza o *driver* JDBC alterado; e
    - Armazenar os dados das conexões que utilizam o mecanismo Grão e criar um serviço que verifique e elimine, periodicamente, as conexões ao banco de dados destino que não façam parte do mecanismo Grão.

6.3 Diagramas

Um diagrama, na linguagem unificada UML (*Unified Modeling Language*), é uma representação gráfica de um conjunto de elementos (classes, interfaces, colaborações, componentes, nós, por exemplo) e são usados para visualizar o sistema sob diferentes perspectivas. A UML define um número de diagramas que permite dirigir o foco para aspectos diferentes do sistema de maneira independente. Seu principal objetivo é facilitar a compreensão do sistema que está sendo desenvolvido (Vargas, 2008). Um diagrama de componentes UML é um gráfico de componentes ligados por relações de dependência entre as classes de um sistema e seus respectivos elementos. Este diagrama facilita o entendimento da organização, das dependências de um sistema e de informações escondidas (Booch et al, 2001).

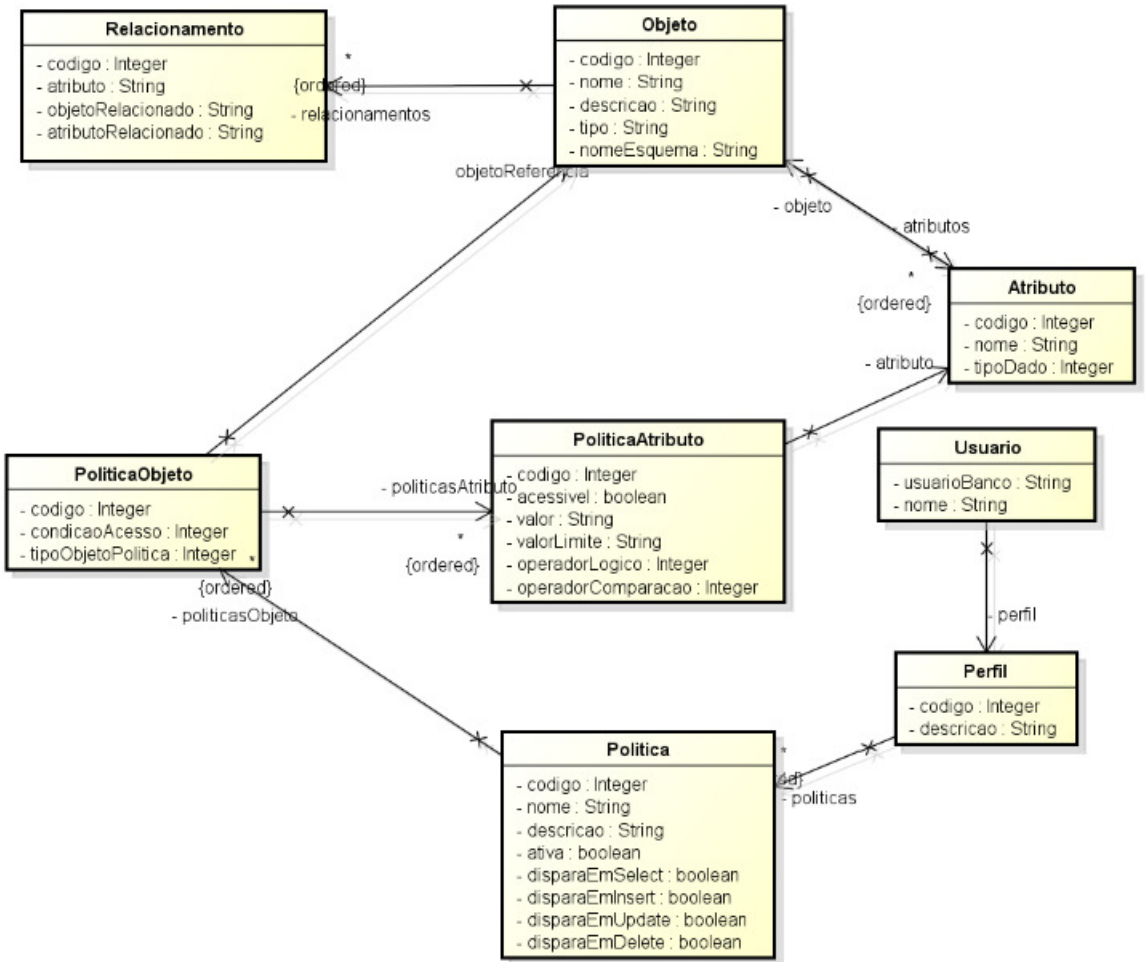
O Diagrama Entidade Relacionamento é a notação diagramática associada ao modelo entidade relacionamento. Há muitas notações diagramáticas alternativas para a exibição de diagramas ER. Uma delas é a notação da *Universal Modeling Language* (UML — Linguagem de Modelagem Universal) para os diagramas de classe, os quais foram propostos como padrão para a modelagem conceitual de objetos (Elmasri e Navathe, 2005).

Para facilitar o entendimento do protótipo criado, serão detalhados os digramas de entidade relacionamento e de componentes nas seções 6.3.1 e 6.3.2.

6.3.1 Diagrama de Entidade Relacionamento

A Figura 6.2 exibe o diagrama entidade relacionamento utilizado pelos módulos Policy Forcer e Policy Manager:

Figura 6.2 Diagrama Entidade Relacionamento da Ferramenta Grão.



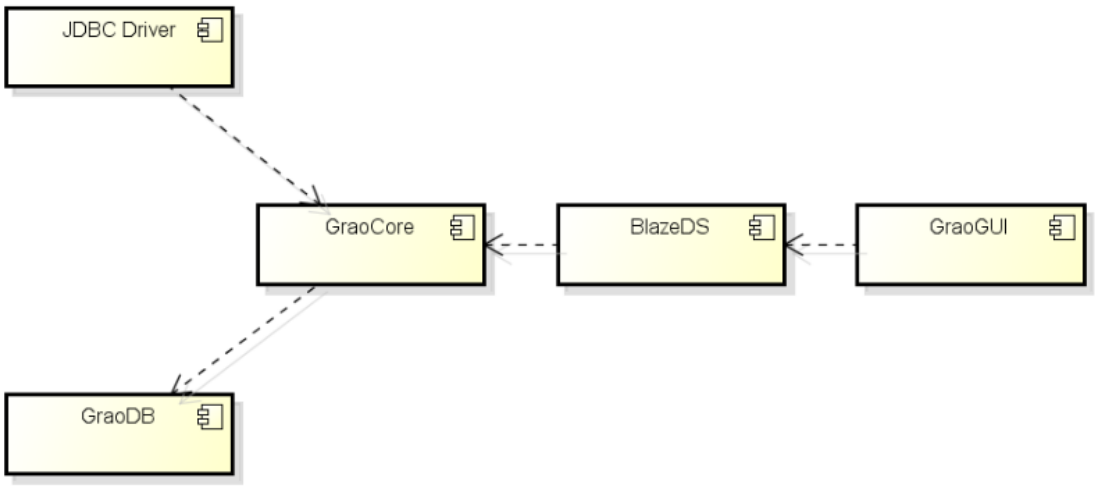
Fonte: Elaborado pelo autor.

No Anexo D será detalhada a funcionalidade de cada uma das entidades e atributos exibidos na Figura 6.2.

6.3.2 Diagrama de Componentes

A Figura 6.3 exibe o diagrama de componentes da ferramenta Grão:

Figura 6.3 Diagrama de Componentes da ferramenta Grão.



Fonte: Elaborado pelo autor.

Os componentes exibidos na Figura 6.3 são descritos abaixo:

- JDBC Driver - *Driver* padrão JDBC com alterações para chamada do método aplicaPolitica do GraoCore. A classe alterada no *driver* é a classe que implementa a interface Statement, responsável pela execução dos comandos SQL;
- GraoCore - Responsável por processar os comandos SQL enviados pelo usuário e, baseado nas políticas cadastradas, modificar o comando para que o usuário de forma transparente só acesse os dados a que tem permissão;
- GraoDB – Sistema de Banco de Dados embarcado H2 acessado tanto pelo GraoCore para recuperar as políticas previamente cadastradas quanto pelo componente GraoGUI para armazenar as regras de política;
- GraoGUI - Componente visual para cadastramento de objetos, usuários, perfis e regras das políticas de acesso aos dados; e
- BlazeDS - Responsável pela conversão de objetos *flex* (GUI) para os objetos Java (GraoCore).

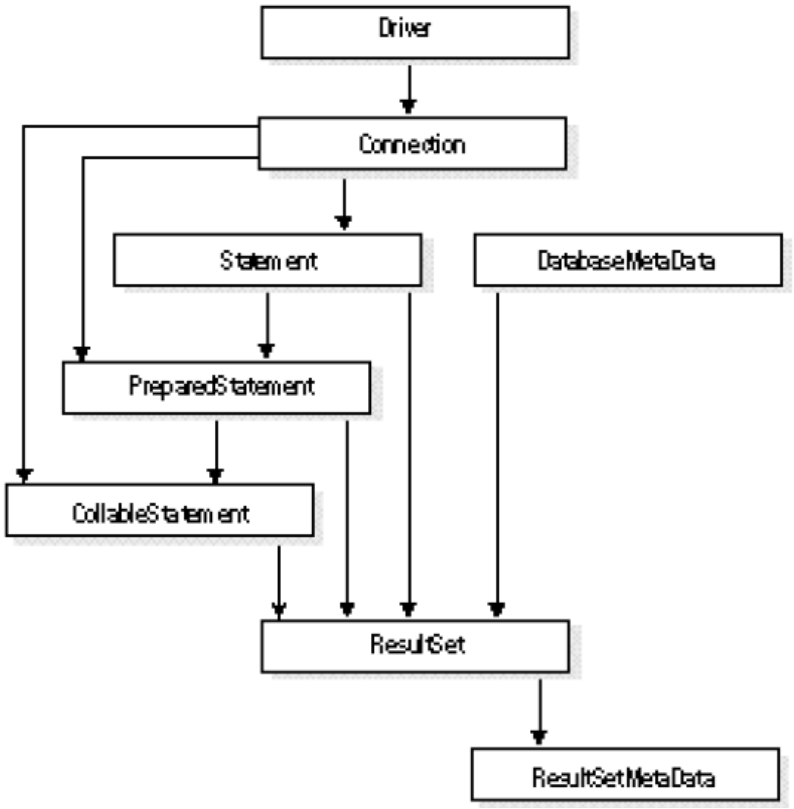
6.4 Tecnologias Adotadas

Foram utilizadas tecnologias atuais e que estivessem sendo utilizadas na construção de aplicações Web. Nas próximas subseções elas serão descritas bem como a associação das tecnologias utilizadas com os módulos e componentes descritos anteriormente.

6.4.1. JDBC

JDBC é uma API, sigla para *Application Program Interface*, que permite embutir comandos SQL como argumentos aos métodos das interfaces definidas. No entanto, cada fornecedor de SGBD deve prover a implementação de tais interfaces. Tal implementação guia as chamadas dos comandos SQL de forma que o fornecedor do SGBD reconheça e trate da maneira adequada (Reese, 2000). As interfaces do *driver* JDBC são visualizadas através da Figura 6.4.

Figura 6.4 Interfaces do Driver Jdbc.



Fonte: Patel e Moss, 1997.

A linguagem SQL, sigla para *Structured Query Language*, é um padrão ANSI e os maiores fornecedores de sistemas de banco de dados relacionais permitem a execução dos comandos SQL seguindo esta padronização. Além disso, o *driver* JDBC é fortemente baseado e totalmente compatível com o padrão ANSI-92. Desta forma, é possível utilizar o mesmo comando SQL para qualquer sistema de banco de dados relacional, ou seja, não é necessário escrever um programa para o SGBD Oracle (Oracle Corporation, 2011) e outro para o Sybase (Sybase Corporation, 2012), por exemplo (Huwei et al, 1998). A partir do exposto, optou-se por utilizar um mecanismo de alteração da interface Statement do *driver* JDBC (componente JDBC *Driver*) com o objetivo de interceptar e reescrever o comando SQL a partir da política de segurança definida conforme trecho em destaque abaixo:

Código 6.1. Alteração da interface Statement com o objetivo de interceptar e reescrever o comando SQL.

```
public java.sql.ResultSet executeQuery(String sql)
       throws SQLException {
       synchronized (checkClosed()) {

       MySQLConnection locallyScopedConn = this.connection;
       /*
       * Interceptação do comando para tratamento no Grao
       */
       sql = PolicyManager.aplicaPolitica(this.connection.getUser(), sql);

       this.retrieveGeneratedKeys = false;

       resetCancelledState();


       checkNullOrEmptyQuery(sql);
```

6.4.2. MySQL

Conforme citado anteriormente, os testes do protótipo desenvolvido como prova de conceito e os experimentos foram baseados no SGBD MySQL na versão 5.5. Abaixo são listados os fatores que influenciaram a escolha do SGBD MySQL (MySQL, 2012):

- Gratuito;
- *Open source*;

- Multi-usuário; e
- Facilidade de edição e recompilação do *driver* JDBC para este SGDB.

Na Figura 6.1, que detalha a arquitetura do protótipo Grão, a utilização do MySQL é representada pelo componente chamado "D2 - Banco de Dados", ou seja, é o SGBD onde o acesso aos dados precisa ser controlado e onde a aplicação realiza os comandos SQL.

6.4.3. Sistema de Banco de Dados Embarcado – H2

Um sistema de banco de dados embarcado é um sistema de banco de dados que reside em um sistema embarcado. No entanto, um sistema de banco de dados embarcado está escondido dentro da aplicação e não está visível ao usuário da referida aplicação. Nos sistemas de bancos de dados tradicionais (relacionais), as principais características oferecidas são escalabilidade, flexibilidade, funcionalidades, por exemplo. Já nos embarcados, características como tamanho e utilização de recursos (processador, memória, entre outros) são fatores críticos. Por este motivo, as principais características dos sistemas de bancos de dados embarcados são (Tešanovi´c et al., 2002):

- Redução na alocação de recursos (memória RAM, por exemplo);
- Suporte a vários sistemas operacionais; e
- Alta disponibilidade.

Além das vantagens expostas acima, optou-se por utilizar o sistema de banco de dados embarcado H2 (H2 Database Engine, 2012) na versão 1.3.166 nesta abordagem, apenas para armazenar os dados (repositório) das políticas de seguranças através do módulo Policy Manager (componente GraoDB), pelos motivos abaixo:

- Não necessita de instalação de um SGBD no servidor de aplicação onde residirá o *driver* JDBC;
- Acesso ao banco de dados embarcado restrito ao próprio *driver* / protótipo implementado; e
- Acesso único (sem concorrência).

6.4.3. Java / Flex / BlazeDS

Os módulos Policy Manager e Policy Forcer foram implementados utilizando a linguagem Java (JAVA, 2012) na versão 1.6. O módulo Policy Manager (componente

GraoGUI), responsável pela interface gráfica para gerenciar as políticas de segurança, utilizou o *framework* BlazeDS (BlazeDS Developer Guide, 2012) que é responsável por integrar Adobe Flex (Adobe Flex, 2012) a Java. O Adobe Flex é um *framework open source* utilizado para a construção de *Rich Internet Applications* que contém todas as formas de comunicação com serviços existentes no mercado, como HTTP, web *services* e também *Remoting*. Para que o suporte a *Remoting* seja funcional, é necessário a utilização de um *DataServices* no *back-end*. Um dos melhores e mais maduros é o BlazeDS, uma solução desenvolvida pela própria Adobe (Fraga, 2009). A versão do *framework* utilizada foi a 4.0.

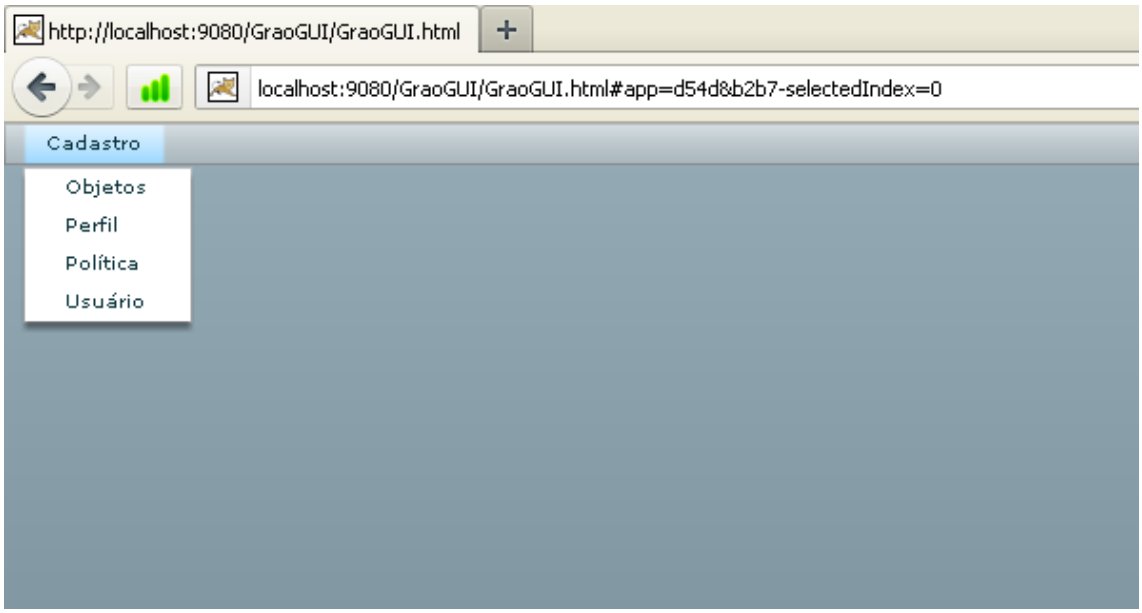
6.4.4. Squirrel SQL

SQuirreL SQL Client é um programa *open source* escrito na linguagem Java que permite a conexão, visualização de dados e execução de comandos SQL em bancos de dados que sejam acessíveis através do *driver* JDBC (SQuirreL SQL Client, 2012). Este programa foi utilizado para realizar os experimentos através do *driver* JDBC customizado com o objetivo de validar a metodologia e a ferramenta Grão. A versão do software utilizada foi a 3.4.0.

6.5 Fluxo Navegacional

A ferramenta Grão possui uma interface gráfica no módulo Policy Manager descrito anteriormente. Tal interface é composta por poucas telas, visto que a inteligência / funcionalidade principal da ferramenta Grão está contida nos módulos Interceptor e Policy Forcer. A funcionalidade principal das telas (módulo Policy Manager) é persistir no sistema de banco de dados embarcado H2 as regras que serão utilizadas pelos demais módulos com o objetivo de forçar a utilização das políticas de segurança definidas. A opção (*menu)* principal da ferramenta Grão é mostrada na Figura 6.5:

Figura 6.5 Tela Principal da Ferramenta Grão.



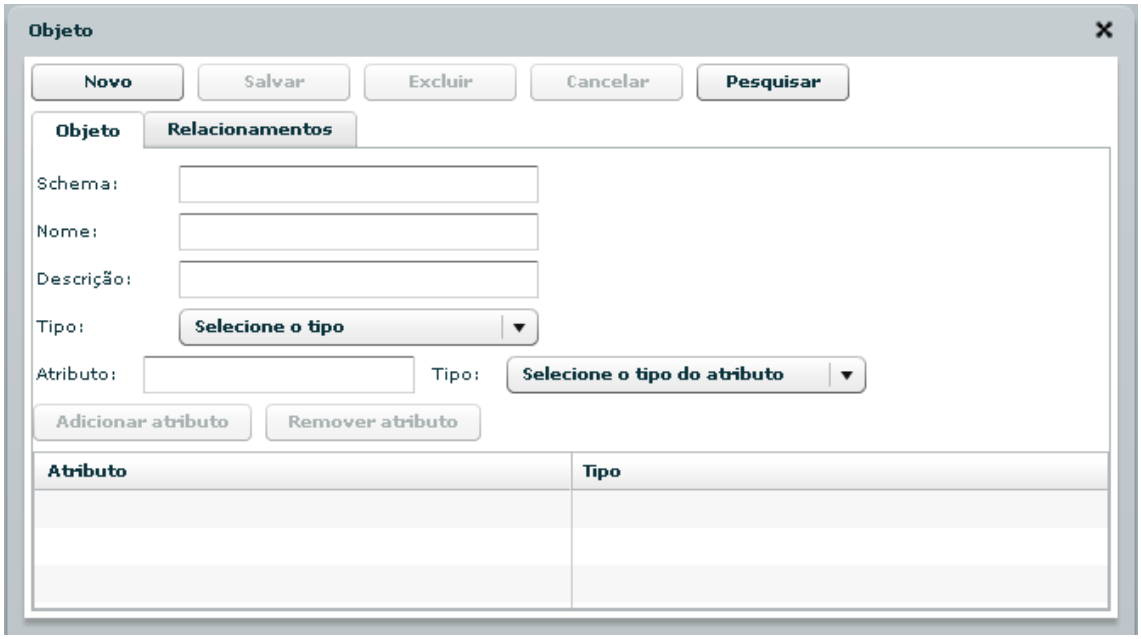
Fonte: Elaborado pelo autor.

O *menu* principal é responsável pela administração dos usuários, perfis, objetos e políticas de segurança. Nas próximas seções serão detalhadas as funções (telas) da ferramenta. Na Seção 6.6 serão exibidos alguns exemplos práticos de como utilizar a ferramenta.

6.5.1 Cadastro de Objetos, Atributos e Relacionamentos

Ao clicar com o mouse na opção “Cadastro”, subopção “Objetos”, é exibida a tela mostrada na Figura 6.6 para cadastro de todos os objetos (tabelas e visões) utilizados como base e como filtro nas políticas que serão criadas. Ao terminar o cadastro de cada objeto é necessário cadastrar, para cada objeto, todos os seus atributos.

Figura 6.6. Cadastro de Objetos e Atributos. Seção Relacionamentos



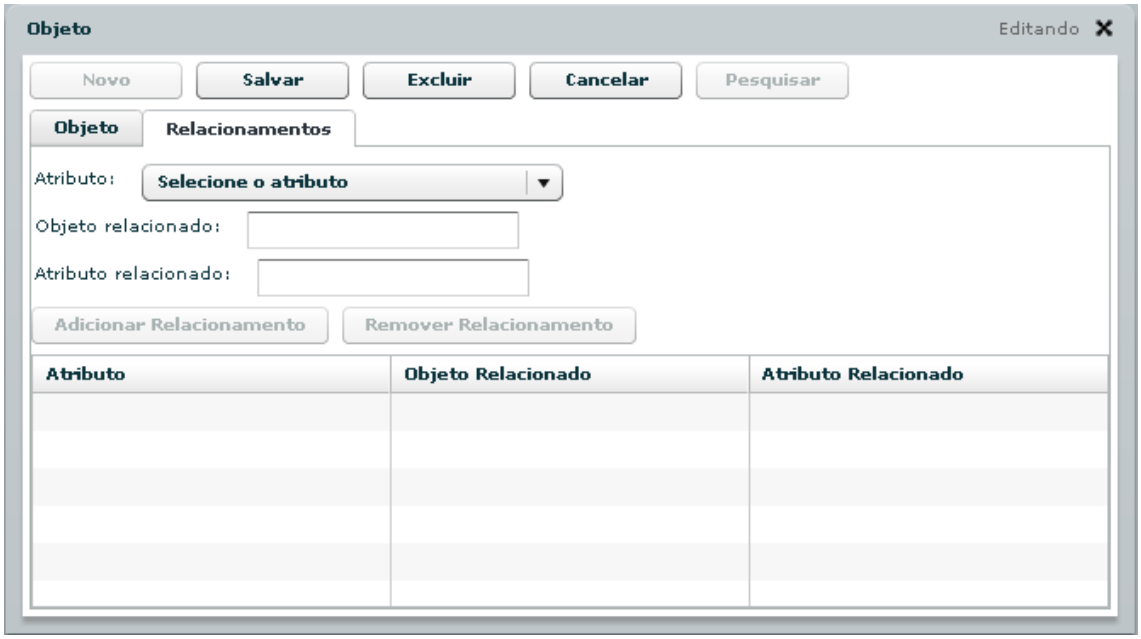
Fonte: Elaborado pelo autor.

Através da mesma tela também é possível definir os "Relacionamentos" entre um ou mais "Objetos" com o objetivo de permitir a definição posterior de políticas de segurança que envolvam mais de um objeto através das condições de junção definidas através desta opção (Figura 6.7).

Para incluir um novo relacionamento deve-se selecionar o objeto que conterá o relacionamento e informar:

- Objeto relacionado;
- Atributo de Relacionamento; e
- Atributo Relacionado.

Figura 6.7. Cadastro de Objetos e Atributos. Seção Relacionamentos.

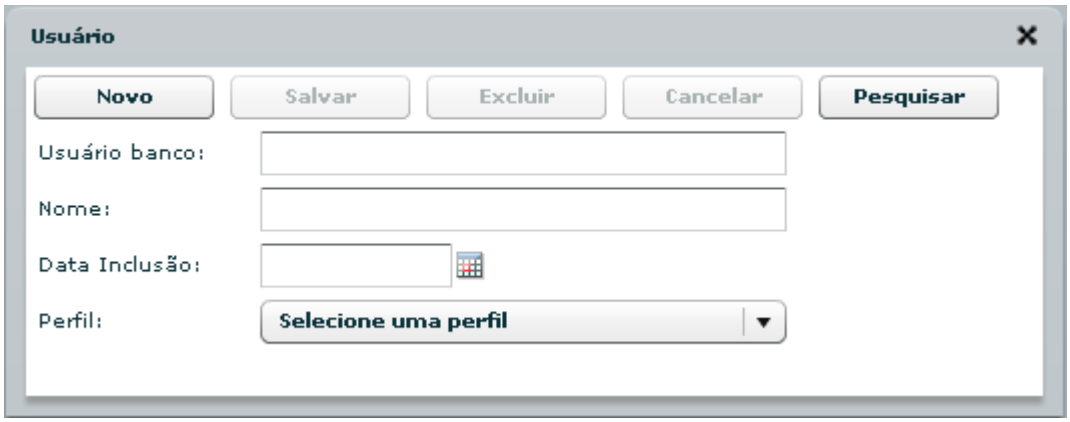


Fonte: Elaborado pelo autor.

6.5.2 Cadastro de Usuários

Ao clicar com o mouse no *menu* "Cadastro", submenu "Usuários", é exibida a tela mostrada na Figura 6.8 para cadastro dos usuários do banco de dados ou aplicação que sofrerão análise de permissão através das políticas que serão criadas.

Figura 6.8. Cadastro de Usuários.



Fonte: Elaborado pelo autor.

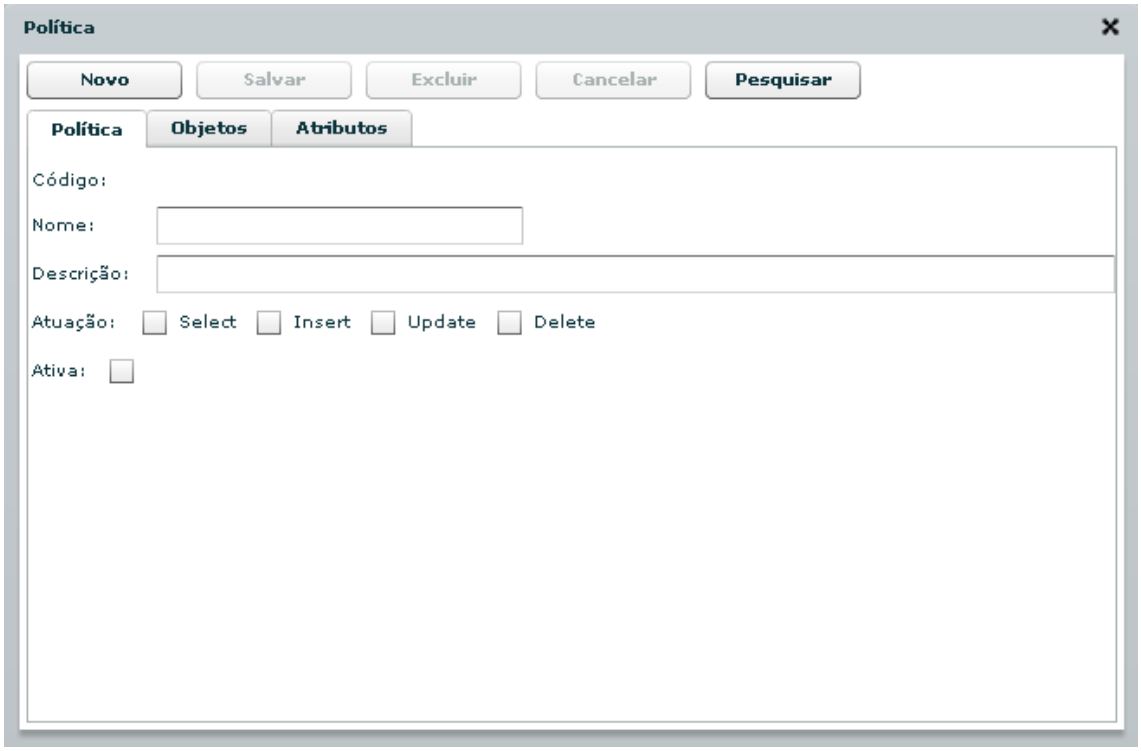
Para incluir um novo usuário deve-se informar:

- Nome do Usuário no Banco de Dados;
- Nome (Descrição) do Usuário Completo;
- Data de Inclusão; e
- Perfil de Acesso.

6.5.3 Cadastro de Políticas

Ao clicar com o mouse no *menu* "Cadastro", submenu "Política", é exibida a tela com três abas conforme visualizado na Figura 6.9. Na aba "Política" é possível realizar o cadastro de todas as políticas de acesso aos dados. Além disso, através dessa função, é possível informar em que diferentes tipos de comandos DML (Insert, Update, Delete) a política será executada e se a política está ativa ou não.

Figura 6.9. Cadastro de Políticas. Seção Política.



Fonte: Elaborado pelo autor.

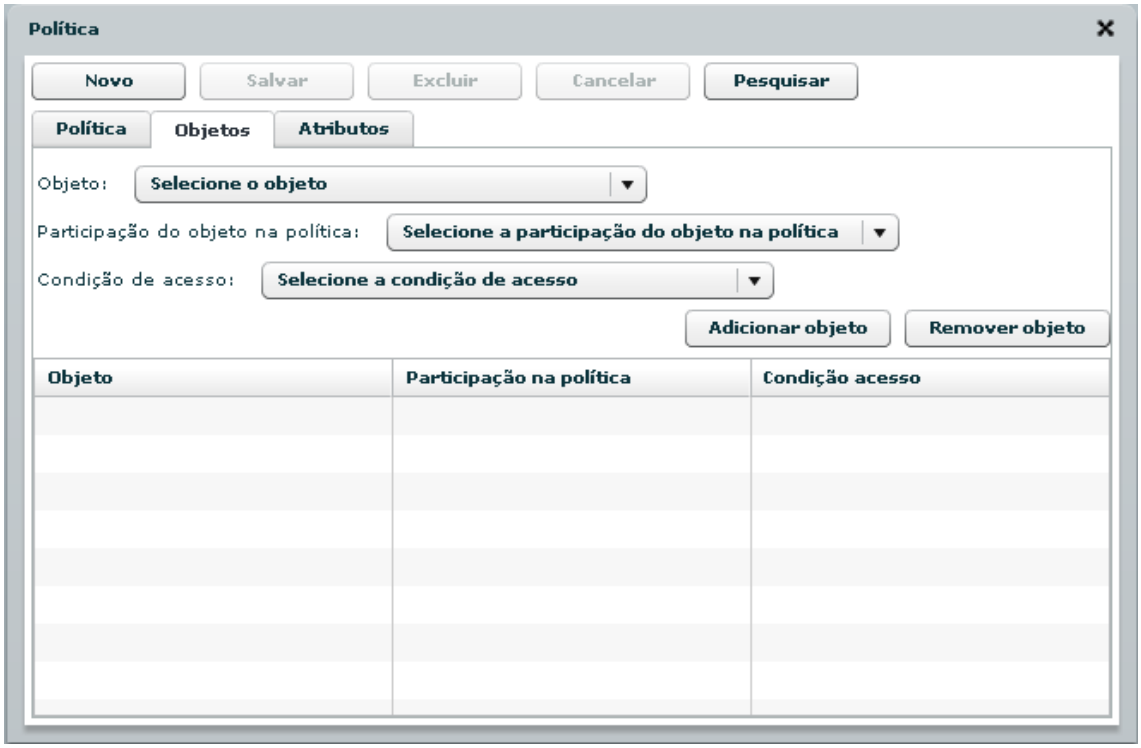
Para incluir uma nova política deve-se informar:

- Nome da Política;
- Descrição da Política;
- Dispara no Select (Sim/Não);

- Dispara no Insert (Sim/Não);
- Dispara no Update (Sim/Não);
- Dispara no Delete (Sim/Não); e
- A política está ativa (Sim/Não).

Na seção “Objetos” deve-se associar os objetos (Tabelas ou Visões) conforme descrito na seção 6.5.1 à política de segurança criada. Para cada objeto associado, deve-se informar qual a participação desse objeto na política. Se o objeto é o principal da política, ou seja, se o filtro será realizado no acesso a esse objeto, será indicado que o objeto possui participação “Principal”. Caso o objeto seja utilizado apenas para relacionar o objeto principal ao objeto que está sendo cadastrado, como forma de filtrar dados a partir de uma tabela relacionada, será indicado que o objeto possui participação “*Lookup*”. Além disso, deve-se escolher a condição de acesso ao referido objeto, ou seja, se o acesso será total (Tudo), nenhum (Nada) ou através de filtros definidos pelos atributos dos objetos associados (Através de atributo). A Figura 6.10 exibe as informações solicitadas na aba “Objetos” da tela para cadastro de políticas.

Figura 6.10. Cadastro de Políticas. Aba Objetos.



Fonte: Elaborado pelo autor.

Para associar objetos a uma política deve-se informar:

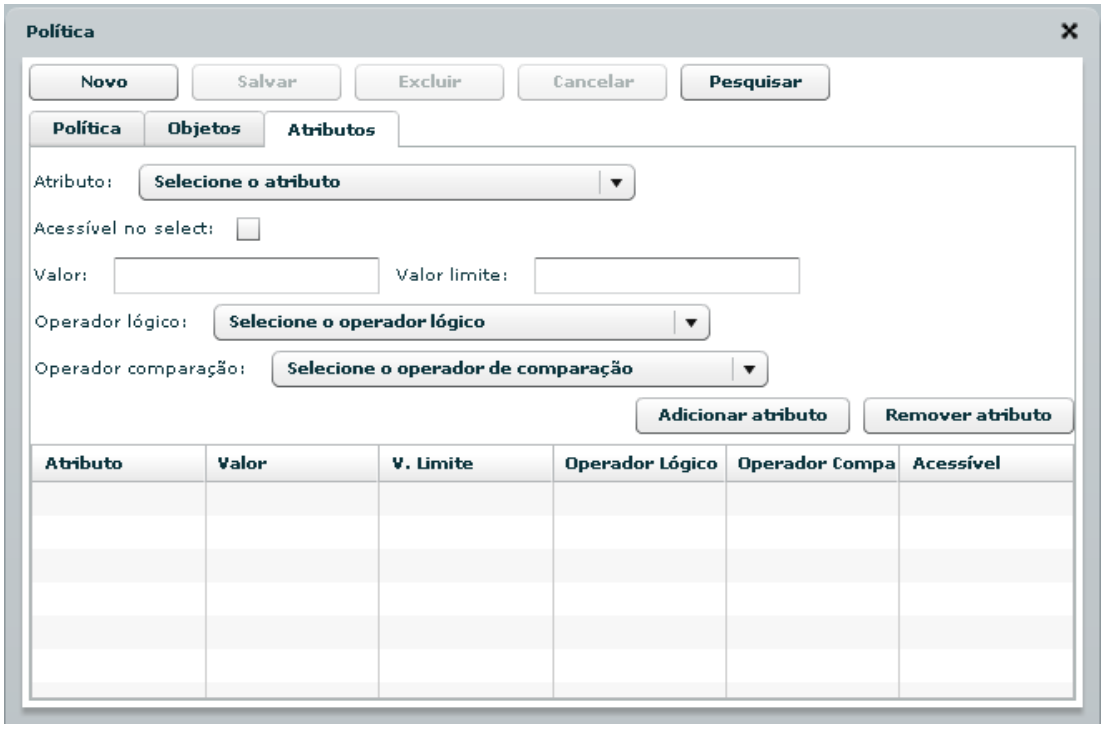
- Nome do Objeto;
- Participação do Objeto na Política (Principal ou *Lookup*); e
- Condição de Acesso (Tudo, Nada ou Através de atributo).

Na aba "Atributos" deve-se associar os objetos (Tabelas ou Visões) cuja condição de acesso definida na aba "Objetos" foi "Através de atributo". Para cada objeto associado com tal condição, deve-se informar:

- Atributo – Atributos referentes ao objeto selecionado na aba "Objetos". Deve-se selecionar o atributo cujo filtro será criado;
- Operador de comparação – Cujos valores possíveis são "Maior", "Menor", "Igual", "Diferente", "Maior ou Igual", "Menor ou igual" ou "Entre";
- Operador lógico – Cujos valores possíveis são "And", "Or" e "Not";
- Acessível no Select - Se o atributo puder ser acessado no caso de um comando SQL Select, deve ser marcada esta opção;
- Valor – Valor para comparação em relação ao Operador de comparação escolhido anteriormente; e
- Valor Limite – Caso o Operador de Comparação escolhido seja o "Entre" é necessário definir um intervalo sendo o campo Valor o limite inferior e o Valor Limite, o limite superior.

A Figura 6.11 exibe as informações solicitadas na aba "Atributos" da tela para cadastro de políticas:

Figura 6.11. Cadastro de Políticas. Seção Atributos.

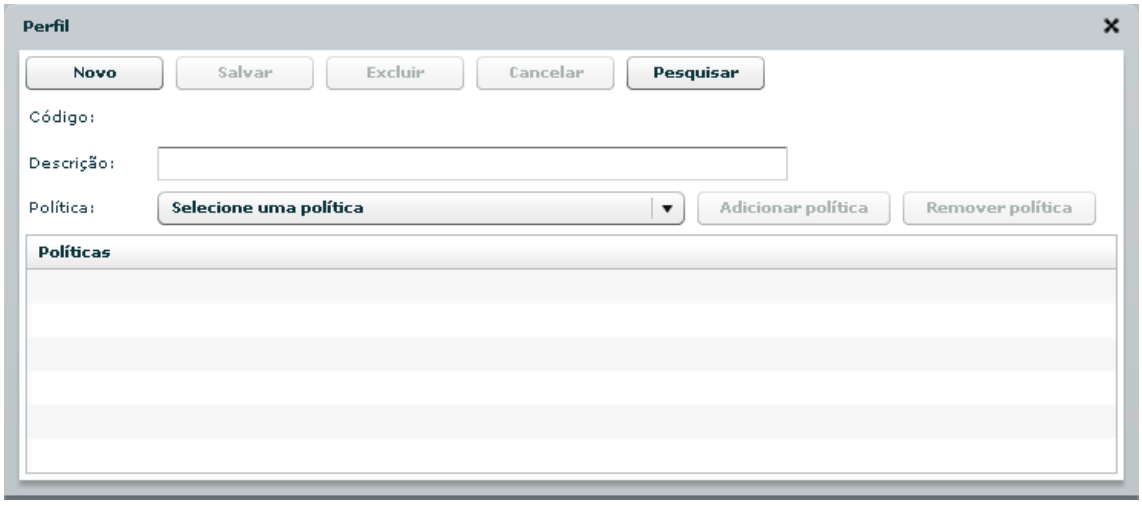


Fonte: Elaborado pelo autor.

6.5.4 Criação de Perfis de Acesso e associação Perfil/Política

Ao clicar com o mouse no *menu* "Cadastro", submenu "Perfis", é exibida a tela mostrada na Figura 6.12 para cadastro dos perfis de usuários no banco de dados. Além disso, através dessa função é possível associar as políticas criadas para cada perfil de usuário.

Figura 6.12. Criação de Perfis de Acesso, associação Perfil/Usuário e Perfil/Política.



Fonte: Elaborado pelo autor.

Para incluir um novo perfil deve-se informar:

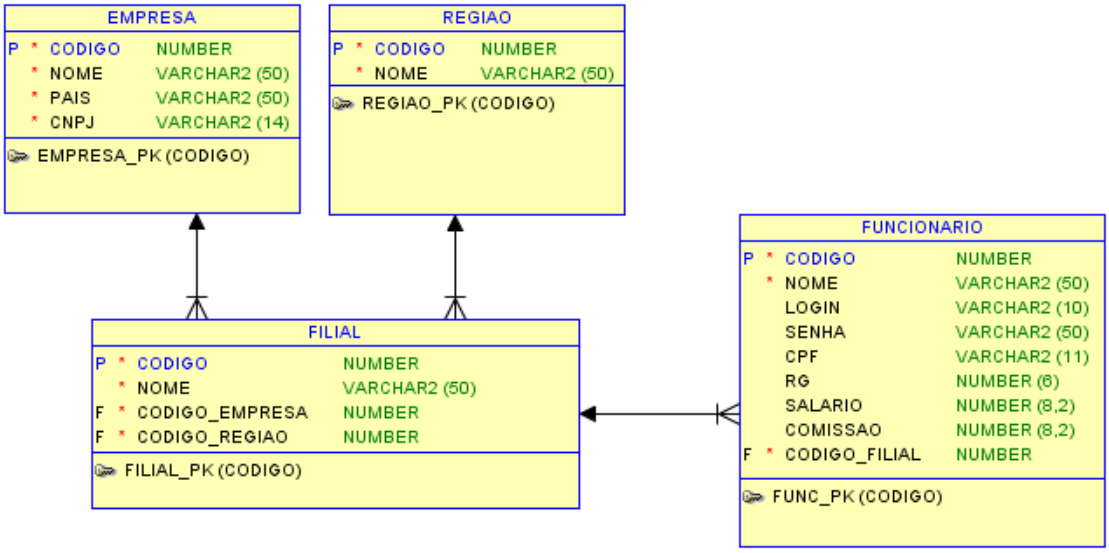
- Código do Perfil; e
- Descrição do Perfil.

Para associar um usuário ao perfil de acesso deve-se escolher a política na lista de políticas cadastradas e clicar no botão “Adicionar política”. Ao final deste processo, todo o ciclo de definição de uma política de segurança foi finalizado.

6.6 Exemplos Práticos de Utilização

Para melhor entendimento do protótipo proposto, o mesmo será aplicado sob uma política que modela um ambiente corporativo fictício. O modelo entidade relacionamento (MER) da Figura 6.13 define parte das entidades e atributos necessários para cadastro de informações para funcionários de uma empresa:

Figura 6.13 Modelo Lógico para Cadastro de Funcionários.



Fonte: Elaborado pelo autor.

A ideia é criar uma política de segurança na tabela FUNCIONARIO para um determinado perfil (nomeado PERFIL1) que pode visualizar informações de todos os funcionários da filial RECIFE que recebam mais que R$ 2500. No entanto, para o mesmo perfil, não é possível visualizar as informações da comissão.

O primeiro passo para se incluir uma política de segurança através da ferramenta Grão é identificar, de acordo com o exemplo descrito, os conceitos explícitos e implícitos ao processo de criação dos perfis de acesso através da metodologia proposta:

- Esquema (*Schema*) – Usuário ou banco de dados proprietário da informação da tabela FUNCIONARIO. Como podem existir várias tabelas de nome FUNCIONARIO em uma mesma instância (mesmo banco de dados), tal conceito é essencial para identificar e direcionar o acesso ao objeto correto. No exemplo utilizado, considerou-se que quem comporta a tabela dos funcionários é o *Schema* TEST;
- Usuário – Conceito do usuário do banco de dados ou aplicação que está tentando acessar a tabela FUNCIONARIO. No exemplo, para facilitar o entendimento, o perfil descrito terá um usuário específico para acessar o banco de dados. Este dado é informado no momento da conexão ao banco de dados. Desta forma, o exemplo prevê a criação de um usuário chamado USUARIO1;
- Comando SQL – Texto contendo o comando SQL utilizado para consultar (*SELECT*) ou realizar operações de manipulação de dados (*INSERT*, *UPDATE* e *DELETE*) em uma tabela ou visão no banco de dados. No exemplo, pode ser simulada a execução do comando:

```
SELECT CODIGO, NOME, SALARIO, COMISSAO
FROM FUNCIONARIO
```

- Perfil de Acesso – Conceito utilizado para agrupar usuários com as mesmas características no acesso aos dados. O exemplo prevê a criação de um perfil chamado PERFIL1;
- Política de Segurança – Conceito que nomeia a política de segurança que está sendo criada. O exemplo prevê a criação de uma política chamada POLITICA1;
- Objeto Principal – Identifica o objeto principal ao qual a política está relacionada para limitar (ou não) o acesso aos dados e colunas. O objeto principal, para o exemplo descrito, é a tabela de FUNCIONARIO;
- Objeto *Lookup* – Identifica os objetos principais que se relacionam (geralmente pela chave estrangeira) com vários objetos *lookup* que devem auxiliar a definição do predicado que será criado e adicionado ao

comando SQL. No exemplo, para o PERFIL1, é necessário relacionar a tabela FUNCIONARIO com a tabela FILIAL para filtrar apenas os funcionários lotados na Filial RECIFE;

- Filtro – Consiste na inclusão, manual, de uma restrição à política que está sendo criada. As restrições podem utilizar operadores lógicos e de comparação. No exemplo, os usuários do PERFIL1 só podem ter acesso aos dados dos funcionários que recebam menos que R$ 2.500 e que façam parte da filial RECIFE. Isto indica que deverá ser realizado um filtro na tabela funcionário conforme exemplo abaixo:

```
SALARIO > 2500
AND CODIGO_FILIAL IN
(SELECT CODIGO
  FROM FILIAL
 WHERE NOME = 'RECIFE')
```

- Predicado – Consiste na montagem da cláusula *WHERE* obtendo a transformação dos filtros e relacionamentos oriundos dos Objetos *Lookup*. Tomando como exemplo o Filtro criado anteriormente, o comando SQL já alterado a partir do predicado criado seria:

```
SELECT CODIGO, NOME, SALARIO, COMISSAO
FROM FUNCIONARIO
WHERE SALARIO > 2500
  AND CODIGO_FILIAL IN
(SELECT CODIGO
  FROM FILIAL
 WHERE NOME = 'RECIFE')
```

- Coluna Invisível – Permite esconder informações específicas dos Objetos Principais. Tomando como base o Comando SQL anterior e a restrição à coluna COMISSAO imposta pela política de segurança, é possível tornar a coluna COMISSAO invisível da seguinte maneira:

```
SELECT CODIGO, NOME, SALARIO, NULL COMISSAO
FROM FUNCIONARIO
```

O segundo passo para a criação de uma política de segurança é identificar, de acordo com o exemplo descrito, quais valores devem ser utilizados para cada um dos

conceitos previamente definidos para que os comandos SQL obedeçam às regras citadas no exemplo para o perfil criado. O Quadro 6.1 detalha, para o perfil, os valores que devem ser utilizados para implementar o exemplo de política citado para o modelo do cadastro de funcionário:

Quadro 6.1 Política para o PERFIL 1.

Conceito	Valor
Perfil	PERFIL1
Política	POLITICA1
Objeto Principal	FUNCIONARIO
Objetos *Lookup*	FILIAL
Coluna Invisível	COMISSAO
Filtros	`SALARIO > 2500`
SQL Original	`SELECT CODIGO, NOME, SALARIO, COMISSAO` `FROM FUNCIONARIO`

Predicado	`WHERE SALARIO > 2500 AND CODIGO_FILIAL IN`
	`(SELECT CODIGO`
	`  FROM FILIAL`
	`  WHERE NOME = 'RECIFE')`

SQL Modificado	`SELECT CODIGO, NOME, SALARIO, NULL AS COMISSAO`
	`FROM FUNCIONARIO`
	`WHERE SALARIO > 2500`
	`  AND CODIGO_FILIAL IN`
	`(SELECT CODIGO`
	`  FROM FILIAL`
	`  WHERE NOME = 'RECIFE')`

Os filtros, predicados e os comandos SQL modificados descritos acima serão forçados pelo engenho chamado de *Policy Forcer*, descrito na Seção 6.2. Após a intervenção deste engenho no comando SQL original, o predicado é formado a partir das políticas definidas. Posteriormente, o comando SQL é alterado adicionando-se o predicado e tratamento das colunas invisíveis. Por fim, o comando SQL alterado é executado no banco de dados e seu resultado será devolvido, de maneira transparente, para a aplicação que o executou finalizando o processo.

O módulo Policy Manager é utilizado para transformar as políticas de segurança em metadados que são utilizados para recuperar o predicado a ser utilizado. Para criar uma política de segurança através da ferramenta Grão é necessário realizar, de maneira simplificada, quatro etapas:

1. Cadastro dos Objetos, Atributos e Relacionamentos envolvidos na política;
2. Cadastro da Política de Segurança e definição da regra de acesso a partir dos Objetos e Atributos definidos;
3. Cadastro do Perfil e associação do Perfil à Política de Segurança; e
4. Criação do usuário e associação ao Perfil de Acesso correspondente.

Na Seção 6.6.1 é descrita a forma de cadastrar a política de segurança descrita no Quadro 6.1 seguindo as quatro etapas descritas acima através da interface gráfica do protótipo Grão. Para facilitar o entendimento do exemplo, o Quadro 6.2 ilustra os dados inseridos nas tabelas FUNCIONARIO e FILIAL utilizados como massa de teste para a execução do experimento descrito. O resultado esperado para o PERFIL1 é delimitado através da marcação do "X" na coluna PERFIL1:

Quadro 6.2 Dados das tabelas FUNCIONARIO e FILIAL e a visualização das linhas para o PERFIL1 (Marcadas com 'X')

FUNCIONARIO					
CODIGO	NOME	SALARIO	COMISSAO	CODIGO_FILIAL	PERFIL 1
1	MAURICIO	5500	100	1	X
2	GUSTAV	4500	0	1	X
3	MARIA	6500	200	1	X
4	PEDRO	3000	0	2	
5	JOAO	5100	100	2	
6	ANA	2000	0	1	

FILIAL	
CODIGO	NOME
1	RECIFE
2	OLINDA
3	NATAL

6.6.1 Criação da Política de Segurança para o PERFIL1

É possível realizar o cadastro da Política 1, descrita no Quadro 6.1, pela ferramenta Grão através dos seguintes passos:

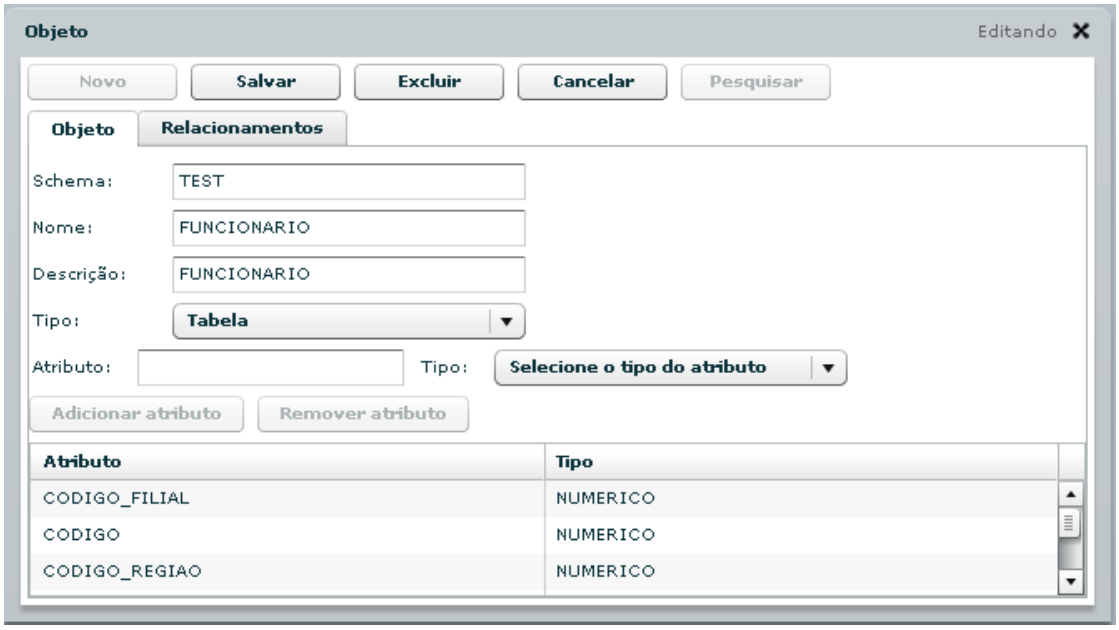
1. Cadastro do Objeto FILIAL e dos seus respectivos Atributos conforme a Figura 6.14:

Figura 6.14 Cadastro do Objeto FILIAL e respectivos Atributos.

Fonte: Elaborado pelo autor.

2. Cadastro do Objeto FUNCIONARIO e dos seus respectivos Atributos conforme a Figura 6.15:

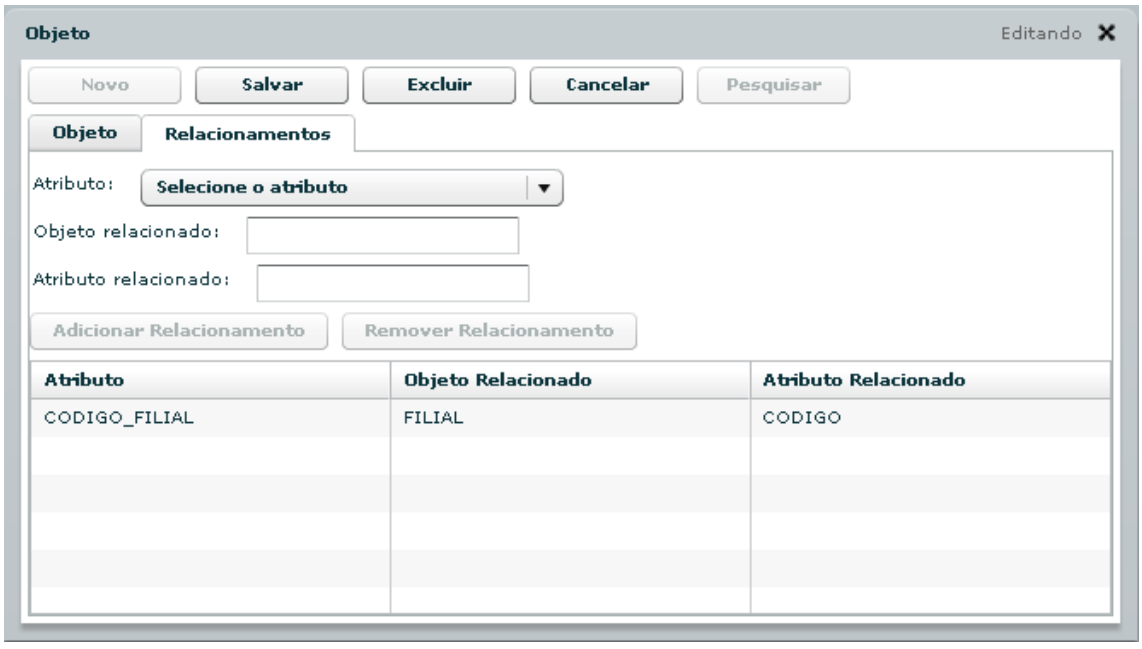
Figura 6.15 Cadastro do Objeto FUNCIONARIO e respectivos Atributos.



Fonte: Elaborado pelo autor.

3. Cadastro do Relacionamento entre os objetos FUNCIONARIO e FILIAL conforme a Figura 6.16:

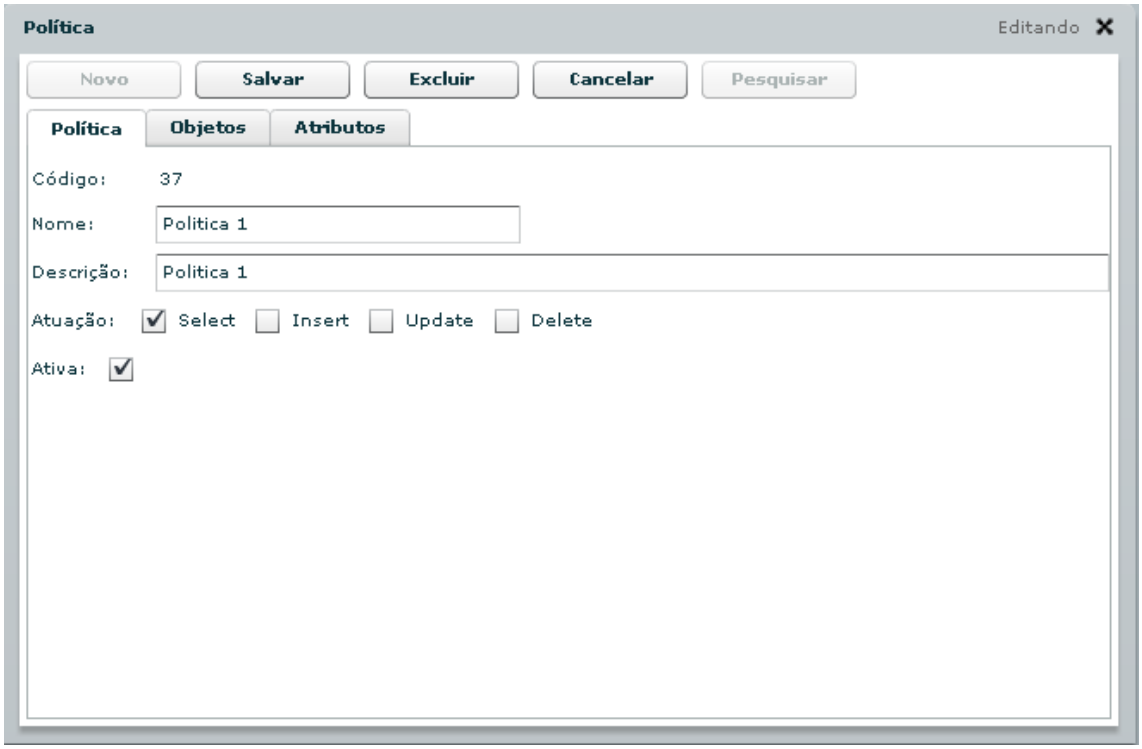
Figura 6.16 Cadastro do Relacionamento entre os Objetos FUNCIONARIO e FILIAL.



Fonte: Elaborado pelo autor.

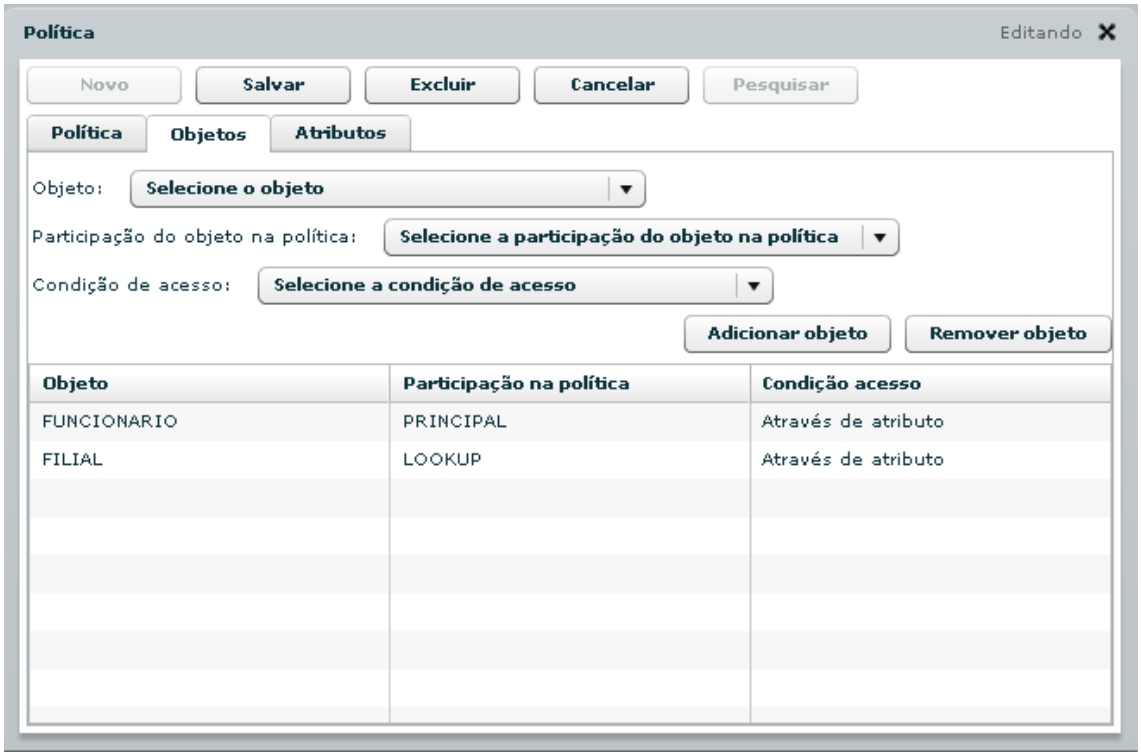
4. Cadastro da Política de Segurança POLITICA1 e definição da regra de acesso a partir dos Objetos e Atributos definidos (visualizar informações de todos os funcionários da empresa em que trabalha que recebam mais que R$ 2.500 e que estejam lotados na filial RECIFE) conforme Figuras 6.17, 6.18 e 6.19. Na Figura 6.20 é definida a forma de acesso a partir da tabela *Lookup* FILIAL:

Figura 6.17 Cadastro da Política POLITICA1. Seção Política.



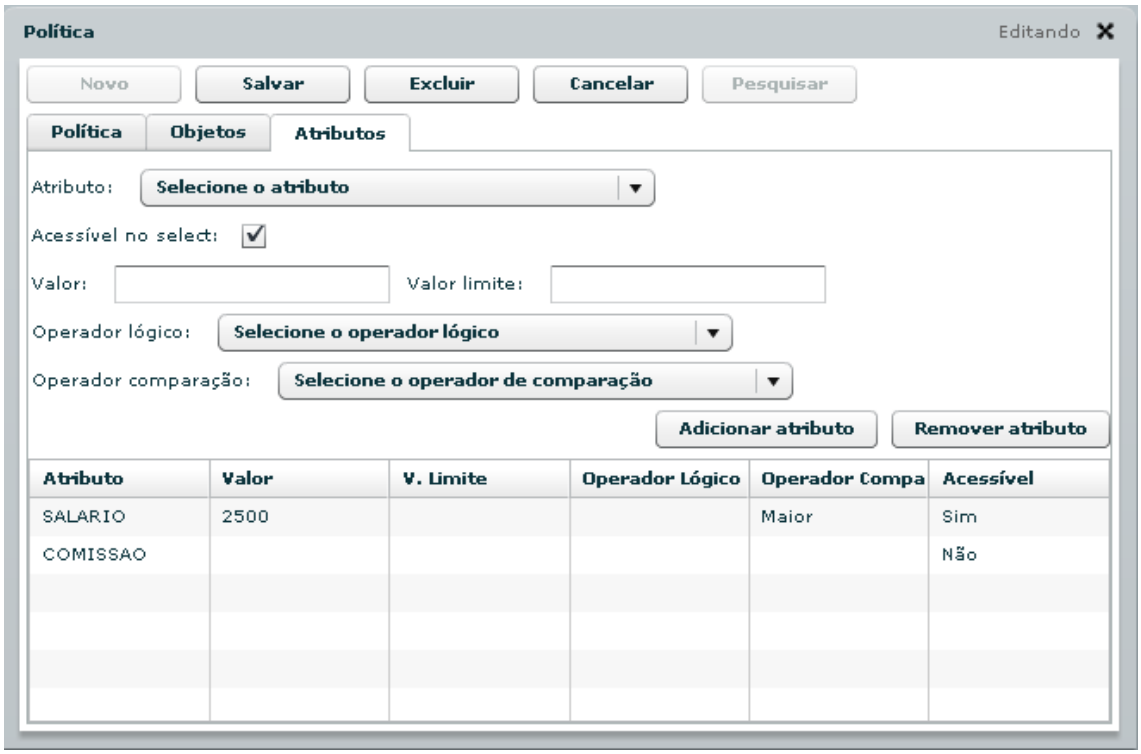
Fonte: Elaborado pelo autor.

Figura 6.18 Cadastro da Política POLITICA1. Seção Objetos.



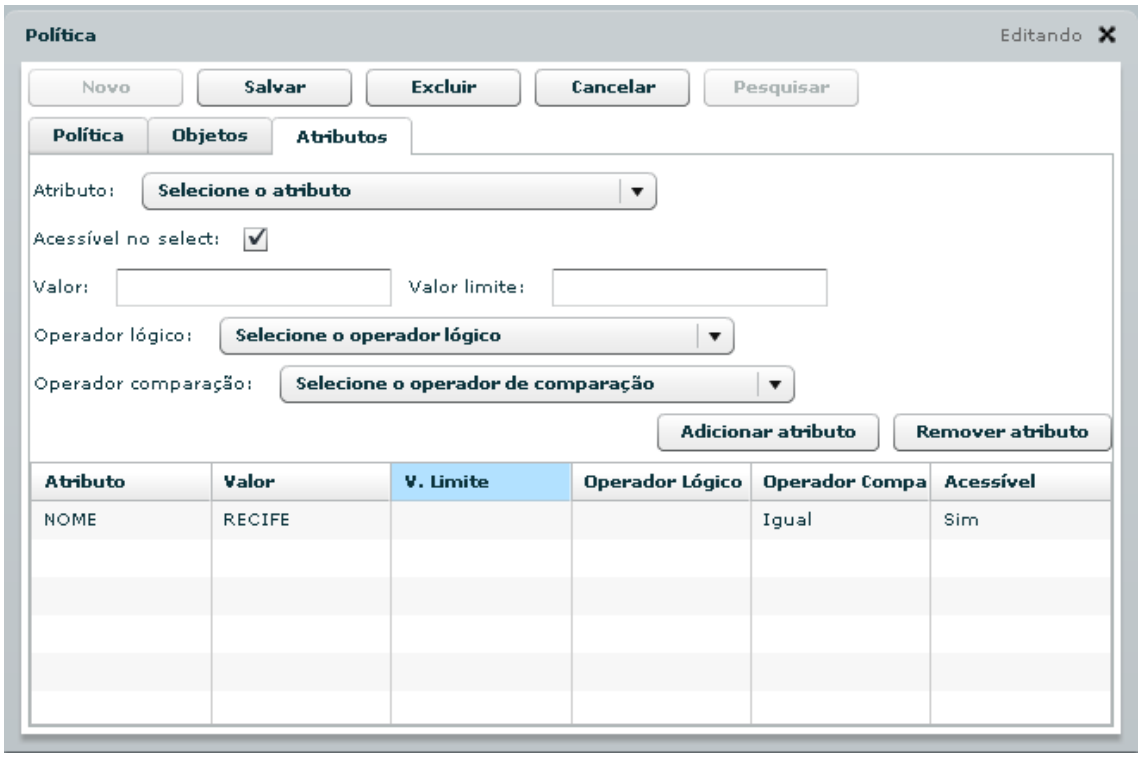
Fonte: Elaborado pelo autor.

Figura 6.19 Cadastro da Política POLITICA1. Seção Atributos do Objeto Funcionário.



Fonte: Elaborado pelo autor.

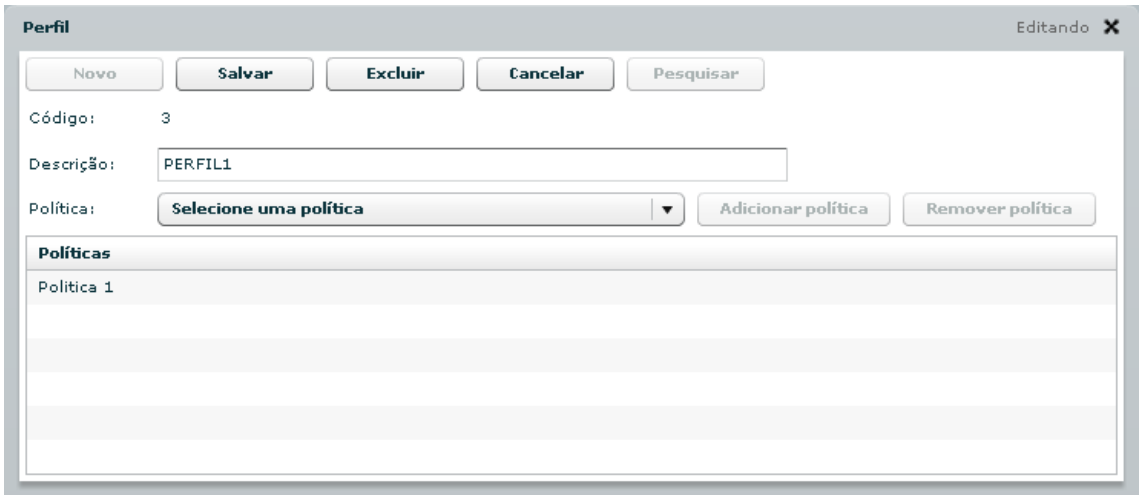
Figura 6.20 Cadastro da Política POLITICA1. Seção Atributos do Objeto Filial.



Fonte: Elaborado pelo autor.

5. Cadastro do Perfil PERFIL1 e associação do PERFIL1 à Política de Segurança POLITICA1 conforme a Figura 6.21:

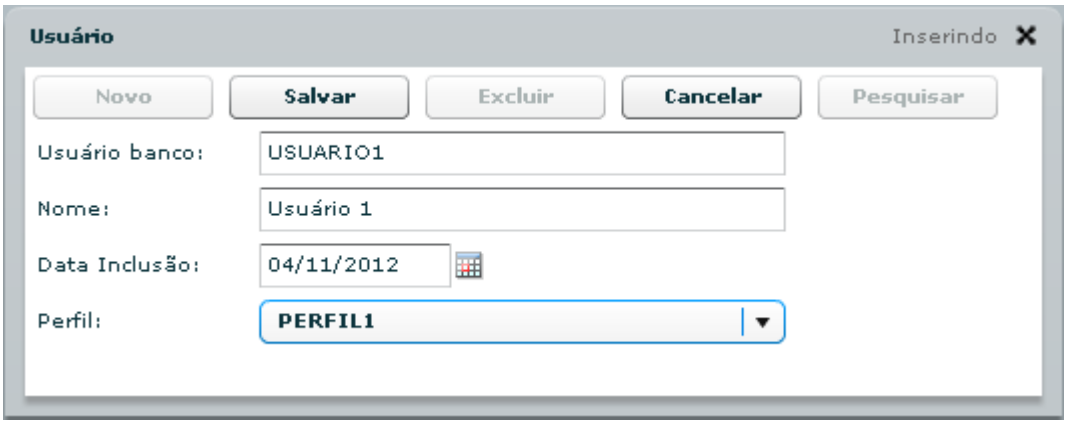
Figura 6.21 Cadastro do Perfil PERFIL1 e associação à Política POLITICA1.



Fonte: Elaborado pelo autor.

6. Criação do usuário USUARIO1 e associação ao Perfil de Acesso PERFIL1 conforme a Figura 6.22:

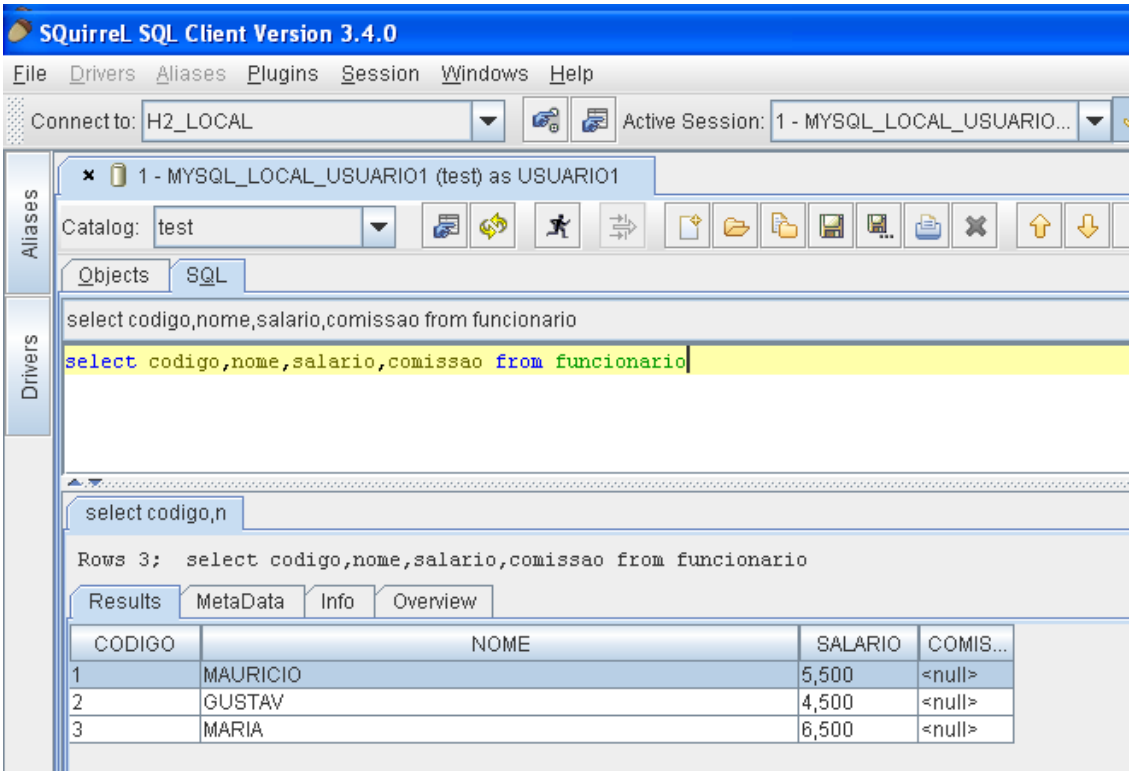
Figura 6.22 Cadastro do Usuário USUÁRIO1 e associação ao Perfil PERFIL1.



Fonte: Elaborado pelo autor.

Cadastrada a Política de Segurança, faz-se necessário validar o resultado do comando SQL Original descrito no Quadro 6.2 executado através do USUARIO1. Para realizar o teste, optou-se por utilizar a ferramenta Squirrel conectando-a ao sistema de banco de dados MySQL utilizando o *driver* JDBC do Grão. A Figura 6.23 descreve o teste realizado e o retorno do comando SQL. Nota-se a restrição ao número de linhas e aos dados da coluna COMISSAO, cujo retorno é nulo para todas as linhas consideradas.

Figura 6.23 Retorno do SELECT para o usuário USUARIO1 (PERFIL1).



Fonte: Elaborado pelo autor.

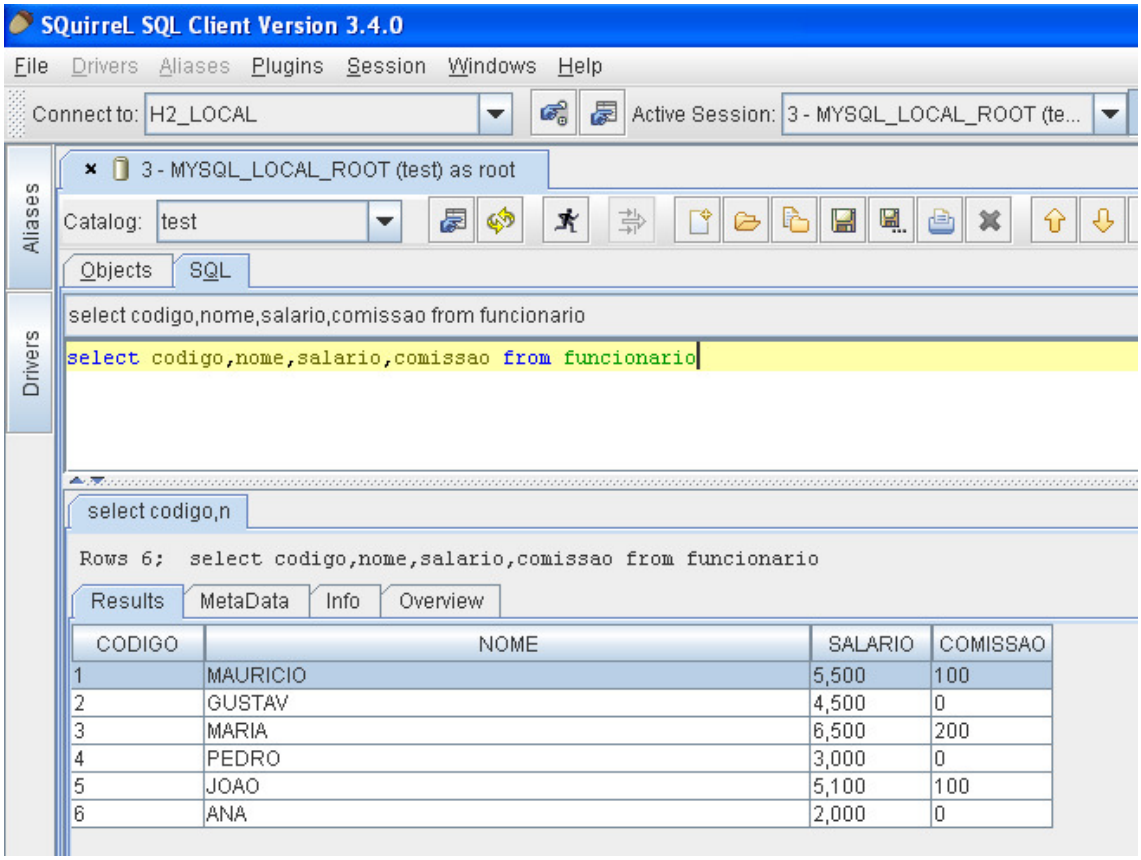
Através do arquivo de *log* do Squirrel visualizado no Código 6.2 constata-se a atuação do mecanismo Grão e a atuação da política de segurança definida interceptando e alterando o comando SQL enviado.

Código 6.2. Trecho de Arquivo de *Log* da ferramenta Squirrel detalhando a atuação do mecanismo Grão para o USUARIO1.

```
INFO com.grao.core.PolicyManager - usr: Usuario 1
INFO com.grao.core.PolicyManager - Usuario possui 1 politicas.
INFO com.grao.core.PolicyManager - Comando para parse: SELECT
CODIGO,NOME,SALARIO,COMISSAO FROM FUNCIONARIO
INFO com.grao.core.PolicyManager - A politica se aplica ao objeto
INFO com.grao.core.PolicyManager - Comando com politica: SELECT
CODIGO,NOME,SALARIO, NULL AS COMISSAO FROM FUNCIONARIO WHERE 1=1  AND
FUNCIONARIO.SALARIO > 2500 AND FUNCIONARIO.CODIGO_FILIAL IN (SELECT
FILIAL.CODIGO FROM FILIAL WHERE 1=1  AND FILIAL.NOME = 'RECIFE' )
INFO com.grao.core.PolicyManager - Politicas aplicadas (60 ms), comando
gerado: SELECT CODIGO,NOME,SALARIO, NULL AS COMISSAO FROM FUNCIONARIO WHERE
1=1  AND FUNCIONARIO.SALARIO > 2500 AND FUNCIONARIO.CODIGO_FILIAL IN (SELECT
FILIAL.CODIGO FROM FILIAL WHERE 1=1  AND FILIAL.NOME = 'RECIFE' )
```

Com o objetivo de comprovar o resultado do experimento, executou-se a mesma consulta, na mesma base de dados, na mesma ferramenta (Squirrel) a partir de um usuário que não sofre ação de Políticas de Segurança (usuário *root*), conforme visualizado na Figura 6.24.

Figura 6.24 Retorno do SELECT para o usuário *root*.



Fonte: Elaborado pelo autor.

Através do arquivo de *log* do Squirrel visualizado no Código 6.3 constata-se a atuação do mecanismo Grão e, como o usuário *root* não possui políticas associadas, não há alteração do comando SQL enviado.

Código 6.3. Trecho de Arquivo de *Log* da ferramenta Squirrel detalhando a atuação do mecanismo Grão para o *root*.

```
INFO com.grao.core.PolicyManager – Aplicando politicas no comando:
SELECT CODIGO,NOME,SALARIO,COMISSAO FROM FUNCIONARIO
INFO com.grao.core.PolicyManager – Usuario: root
INFO com.grao.core.PolicyManager – Comando: SELECT
CODIGO,NOME,SALARIO,COMISSAO FROM FUNCIONARIO
INFO com.grao.core.PolicyManager – usr: root
INFO com.grao.core.PolicyManager – Usuario possui 0 politicas.
INFO com.grao.core.PolicyManager – Comando para parse: SELECT
CODIGO,NOME,SALARIO,COMISSAO FROM FUNCIONARIOINFO com.grao.core.PolicyManager
– Comando com politica: SELECT CODIGO,NOME,SALARIO,COMISSAO FROM
FUNCIONARIOINFO com.grao.core.PolicyManager – Politicas aplicadas (50 ms),
comando gerado: SELECT CODIGO,NOME,SALARIO,COMISSAO FROM FUNCIONARIO
```

6.7 Experimentos Realizados

Foi realizada uma série de experimentos para determinar o desempenho da metodologia sugerida a partir do protótipo criado. A ferramenta Grão, conforme descrita anteriormente, intercepta o comando SQL e adiciona um predicado de acordo com a política de segurança imposta. Desta forma, os experimentos têm como principal objetivo determinar o impacto (custo) introduzido por este desvio no caminho natural do comando SQL através do *driver* JDBC e não da execução do comando SQL pelo SGBD.

6.7.1 Preparação dos Experimentos

Diante do quadro escolhido para análise, a medida de desempenho escolhida (métrica) foi a análise do tempo de execução do processo de interceptação de comandos SQL SELECT no SGBD MySQL, baseado na execução de experimentos. Cada experimento será composto por um número de coletas (execuções de uma mesma instrução) com o mecanismo de segurança proposto. Os parâmetros escolhidos para os experimentos estão descritos no Quadro 6.3.

Quadro 6.3 Parâmetros escolhidos para os experimentos.

Componente	Configuração
Memória RAM	512Mb (Máquina Virtual)
Sistema Operacional	Windows XP (Máquina Virtual)
Processador	Intel(R) Core i5 (Máquina Física)
Disco	450Gb
SGBD	MySQL 5.5
Comando SQL	SELECT, UPDATE
Quantidade de Políticas para as tabelas envolvidas no comando	0, 1, 3, 5
Quantidade de Registros das Tabelas	1.000 (pequena) / 100.000 (média) /1.000.000 (grande)
Mecanismo	Grão
Índice na tabela	Sim

Como o principal objetivo é analisar o grau de interferência do mecanismo proposto, optou-se por fixar comandos SQL que utilizem filtros apenas em colunas indexadas e em tabelas pequenas (já que o volume não influencia no tempo do mecanismo). A partir de tais premissas, foram escolhidos os fatores descritos no Quadro 6.4 para análise dos resultados.

Quadro 6.4 Fatores escolhidos para os experimentos.

Componente	Configuração
Comando SQL	SELECT, UPDATE
Quantidade de Políticas envolvidas no comando	0, 1, 3, 5

Conforme descrito anteriormente, a métrica escolhida foi o tempo de execução de instruções SQL com a alteração do *driver* JDBC. Optou-se por esta medida por ela apresentar as seguintes propriedades:

Linearidade parcial - A métrica escolhida varia proporcionalmente apenas para alguns fatores. Exemplo: Se a quantidade de tabelas na cláusula FROM aumenta, o tempo aumenta, mas se o comando variar, o tempo não necessariamente sofrerá alteração;

Confiabilidade - Os dados foram coletados em ambiente real e sem interferência de outros aplicativos. Além disso, não houve manipulação de dados de qualquer natureza para se chegar aos resultados;

Repetibilidade – É possível repetir as coletas e encontrar resultados parecidos, desde que a configuração do ambiente descrito seja mantida;

Facilidade de medida - Foram obtidas de acordo com programas simples criados para esta finalidade; e

Consistência - Devido à variedade de configurações utilizadas para a obtenção dos resultados. De fato, foram utilizadas combinações diferentes de comandos SQL, com número de políticas e com tamanho de tabelas diferentes de um mesmo SGBD rodando no mesmo equipamento.

No entanto, entende-se que a métrica não apresenta independência, pois certamente a execução dos experimentos em outra plataforma de hardware, por exemplo, resultaria em valores distintos para a métrica adotada.

A partir da métrica definida, realizou-se uma série de experimentos para coleta dos dados levando em consideração a utilização de um *driver* JDBC com o mecanismo proposto. A partir dessa premissa, coletaram-se medidas de oito mil experimentos contendo os resultados retornados com e sem a ferramenta Grão no SGBD MySQL para os comandos SQL SELECT e UPDATE em tabelas pequenas cujo número de políticas variou entre (0,1, 3 e 5) para os objetos contidos no comando SQL. Os oito mil experimentos foram executados através do programa GhostMouse, que permite gravar e automatizar uma sequência de cliques (GhostMouse,2012). O Quadro 6.5 lista o elenco de experimentos realizados:

Quadro 6.5 Elenco dos experimentos realizados.

ID	DESCRIÇÃO	Comando SQL	Qtd Políticas	Número de Coletas
1	GRAO_UPDATE_ZERO	UPDATE	0	1000
2	GRAO_UPDATE_UM	UPDATE	1	1000
3	GRAO_UPDATE_TRES	UPDATE	3	1000
4	GRAO_UPDATE_CINCO	UPDATE	5	1000
5	GRAO_SELECT_ZERO	SELECT	0	1000
6	GRAO_SELECT_UM	SELECT	1	1000
7	GRAO_SELECT_TRES	SELECT	3	1000
8	GRAO_SELECT_CINCO	SELECT	5	1000

As políticas de segurança utilizadas para os experimentos foram criadas para atuar sobre a tabela FUNCIONARIO (descrita na seção 6.6). Criaram-se cinco políticas para simular os comandos SQL e coletar os dados para posterior análise. O Quadro 6.6 lista as políticas criadas:

Quadro 6.6 Políticas de segurança criadas para simular a execução dos experimentos.

Política	Descrição	Filtro (Cláusula WHERE)
1	Limitar o acesso aos funcionários cujo salário seja menor que 3000	SALARIO < 3000
2	Limitar o acesso aos funcionários cuja comissão seja maior que 0 (zero)	COMISSAO > 0
3	Limitar o acesso aos funcionários cuja FILIAL seja igual a 1	CODIGO_FILIAL = 1
4	Limitar o acesso aos funcionários cujo código seja igual a 3	CODIGO = 3
5	Limitar o acesso aos funcionários cujo nome seja ANA	NOME ='ANA'

Os comandos SQL, a quantidade e detalhamento das políticas de segurança utilizadas para os experimentos são descritos no Quadro 6.7:

Quadro 6.7 Comandos SQL antes e depois da análise das políticas de segurança utilizadas no experimento.

ID	Comando SQL	Comando Original	Qtd Políticas	Políticas	Comando Alterado
1	UPDATE	UPDATE FUNCIONARIO SET SALARIO = 5000	0	-	UPDATE FUNCIONARIO SET SALARIO = 5000
2	UPDATE	UPDATE FUNCIONARIO SET SALARIO = 5000	1	1	UPDATE FUNCIONARIO SET SALARIO = 5000 WHERE SALARIO < 3000
3	UPDATE	UPDATE FUNCIONARIO SET SALARIO = 5000	3	1,2,3	UPDATE FUNCIONARIO SET SALARIO = 5000 WHERE SALARIO < 3000 AND COMISSAO > 0 AND CODIGO_FILIAL = 1
4	UPDATE	UPDATE FUNCIONARIO SET SALARIO = 5000	5	1,2,3,4,5	UPDATE FUNCIONARIO SET SALARIO = 5000 WHERE SALARIO < 3000 AND COMISSAO > 0 AND CODIGO_FILIAL = 1 AND CODIGO = 3 AND NOME = 'ANA'
5	SELECT	SELECT NOME FROM FUNCIONARIO	0	-	SELECT NOME FROM FUNCIONARIO
6	SELECT	SELECT NOME FROM FUNCIONARIO	1	1	SELECT NOME FROM FUNCIONARIO WHERE SALARIO < 3000
7	SELECT	SELECT NOME FROM FUNCIONARIO	3	1,2,3	SELECT NOME FROM FUNCIONARIO WHERE SALARIO < 3000 AND COMISSAO > 0 AND CODIGO_FILIAL = 1
8	SELECT	SELECT NOME FROM FUNCIONARIO	5	1,2,3,4,5	SELECT NOME FROM FUNCIONARIO WHERE SALARIO < 3000 AND COMISSAO > 0 AND CODIGO_FILIAL = 1 AND CODIGO = 3 AND NOME = 'ANA'

6.7.2 Ameaças à validade dos Experimentos Realizados

Os experimentos apesar de terem sido realizados com o máximo de cuidado e rigor possíveis, podem ter sido influenciados pelas seguintes ameaças:

- Ambiente livre de concorrência – Os experimentos foram realizados utilizando uma única conexão ao banco de dados;
- Comandos SQL envolvendo poucas tabelas - Os experimentos foram realizados utilizando junções simples entre duas tabelas (no máximo); e
- Tamanho do cadastro de políticas de segurança, usuários e perfis de acesso – Os experimentos foram realizados sobre um repositório que

continha uma pequena quantidade de políticas de segurança, usuários e perfis de acesso cadastrados.

6.7.3 Resultados e Análise dos Experimentos Realizados

Para facilitar a análise dos dados de cada experimento definido e, com isso, determinar o grau de interferência do método proposto, ou seja, quanto tempo o mecanismo leva pra buscar e adicionar a cláusula WHERE, criou-se um mecanismo para persistir em um arquivo texto (Código 6. 4) as seguintes informações:

- Comando SQL antes da interceptação;
- Usuário que está executando o Comando SQL;
- Número de Políticas associadas;
- Comando SQL antes da interceptação; e
- Tempo, em milissegundos, do processo de interceptação e alteração do Comando SQL.

Código 6.4 Trecho do arquivo utilizado para coleta de dados do experimento.

```
INFO com.grao.core.PolicyManager - usr: Usuario 1
INFO com.grao.core.PolicyManager - Usuario possui 1 politicas.
INFO com.grao.core.PolicyManager - Comando para parse: SELECT NOME FROM
FUNCIONARIO
INFO com.grao.core.PolicyManager - A politica se aplica ao objeto
INFO com.grao.core.PolicyManager - Comando com politica: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando para parse: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando com politica: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando para parse: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando com politica: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando para parse: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Comando com politica: SELECT NOME FROM
FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
INFO com.grao.core.PolicyManager - Politicas aplicadas (80 ms), comando
gerado: SELECT NOME FROM FUNCIONARIO WHERE 1=1  AND FUNCIONARIO.SALARIO < 3000
```

A partir do arquivo cujo trecho é listado no Código 6.4, foi possível coletar o tempo de interceptação e reescrita (em milissegundos) do Comando SQL para os oito mil experimentos executados.

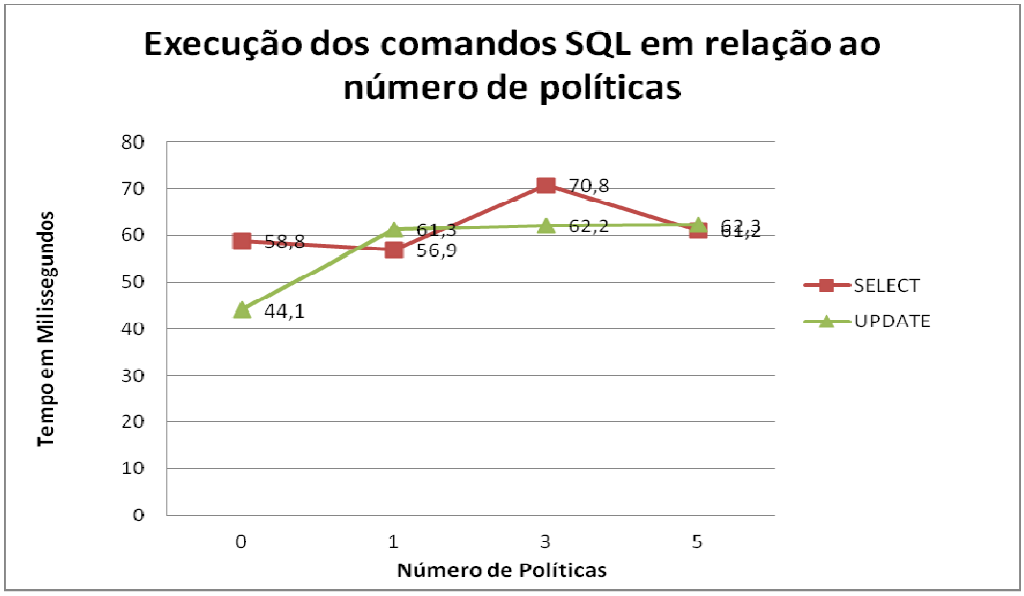
O Quadro 6.8 mostra os resultados obtidos para os experimentos 1 a 8, ou seja, para comandos SQL SELECT e UPDATE, variando o número de políticas com a ferramenta Grão.

Quadro 6.8 Resultados obtidos nos experimentos 1 a 8.

Número de Políticas	Número de Experimentos	Comando SQL	
		SELECT	UPDATE
		Tempo Médio por execução	Tempo Médio por execução
0	1000	58,8	44,1
1	1000	56,9	61,3
3	1000	62,2	70,8
5	1000	62,3	61,2

A Figura 6.25 representa graficamente o resultado dos dados exibidos no Quadro 6.8. A partir desta figura é possível observar o grau de interferência (tempo em milissegundos) na execução dos Comandos SQL em relação ao número de políticas.

Figura 6.25 Tempo Médio (em milissegundos) de interferência na execução dos Comandos SQL em relação ao número de políticas.



Fonte: Elaborado pelo autor.

A partir dos dados coletados pode-se concluir que o nível de interferência do mecanismo proposto é muito baixo comparado ao acesso tradicional (sem utilizar o *driver* JDBC alterado) visto que o tempo médio máximo de interferência coletado foi de 70 milissegundos. Além disso, observa-se que o aumento no número de políticas não é proporcional ao tempo da interferência do mecanismo proposto, ou seja, é possível prever que em um ambiente corporativo, onde o número de políticas é bem maior do que os experimentos realizados, o resultado não sofrerá grandes alterações.

6.8 Limitações do Método e da Ferramenta

Após realizar os testes e validações sobre a ferramenta Grão, foi possível identificar as seguintes limitações:

Limitação de utilização da ferramenta apenas para sistemas ou aplicativos que utilizam conexões JDBC– O método proposto pode ser utilizado apenas quando o mecanismo de conexão ao banco de dados (*driver*) utilizado é o JDBC, ou seja, sistemas que se conectem ao banco de dados de forma nativa ou através de algum outro mecanismo de conexão (ODBC, por exemplo) não podem utilizar a ferramenta proposta. Além disso, o *driver* JDBC utilizado deve permitir alterações em seu código fonte (*open source*).

Perda de Desempenho por conta de ausência de PreparedStatement – Um comando SQL é pré-compilado e armazenado com um objeto chamado PreparedStatement. Este objeto pode ser utilizado para prover desempenho caso o mesmo comando SQL seja executado várias vezes (JavaTM 2 SDK Standard Edition Documentation Version 1.4.2, 2012). No Código 6.5 é descrito um exemplo de utilização do PreparedStatement:

Código 6.5 Exemplo de Utilização do PreparedStatement

```
PreparedStatement pstmt =
con.prepareStatement("UPDATE EMPLOYEESSET SALARY = ? WHERE ID = ?");
   pstmt.setBigDecimal(1, 153833.00)
   pstmt.setInt(2, 110592)
```

No mecanismo proposto, o filtro do comando SQL listado no Código 6.3, ficaria conforme exemplo exibido no Código 6.6:

Código 6.6 Exemplo de comando SQL através do mecanismo proposto

```
"UPDATE EMPLOYEES SET SALARY = ? WHERE ID = 10"
```

Como o mecanismo proposto intercepta e altera o comando SQL utilizando variáveis literais para adicionar o predicado, cada execução de um comando SQL para usuários que possuam políticas de segurança diferentes, será recompilado pelo banco de dados, geralmente diminuindo seu desempenho (The Java Tutorials, 2012).

6.9 Considerações Finais

Apresentou-se neste capítulo a ferramenta Grão, a qual implementa a metodologia proposta por esta dissertação. Descreveu-se sua arquitetura geral, juntamente com o diagrama de componentes e diagrama entidade relacionamento dos principais módulos que a compõem. Além disso, citou-se e justificou-se as tecnologias utilizadas para a construção da ferramenta.

Mostrou-se um exemplo prático de aplicação da metodologia em um ambiente real e apresentou-se os resultados obtidos, validando, desta forma, a eficácia do método.

Para efeito de avaliação da eficiência da aplicação da metodologia, realizou-se uma série de experimentos para identificar o tempo decorrido entre a interceptação do comando SQL e sua reescrita. Por fim, identificaram-se algumas limitações da ferramenta Grão, como por exemplo a utilização da ferramenta apenas para sistemas ou aplicativos que utilizam conexões JDBC e a perda de desempenho por conta de ausência de PreparedStatement.

Capítulo 7: Conclusões e Trabalhos Futuros

Neste capítulo são apresentadas as considerações finais sobre o estudo desenvolvido. Além disso, é descrito uma síntese dos principais resultados obtidos. Também são sugeridos os trabalhos futuros que poderão ter por base esta dissertação.

7.1 Considerações Finais

A pesquisa descrita nessa dissertação abordou aspectos inerentes ao controle de acesso aos dados, em especial ao controle de acesso granular aos dados. Por meio de experimentos, avaliou-se a interceptação dos comandos SQL no nível da camada de conexão com o banco de dados. Propôs-se então, a incorporação de um método capaz de interceptar, analisar os privilégios de acesso e reescrever os comandos SQL. Além disso, tornou-se necessário descrever um método que controlasse o acesso de conexões oriundas de mecanismos diferentes do proposto, com o objetivo de evitar conexões e, consequentemente, acessos não autorizados aos dados. É importante ressaltar que, apesar dos experimentos terem sido direcionados para o SGBD MySQL, o método proposto pode ser utilizado em qualquer SGBD relacional open source, sendo possível também utilizá-lo em bancos de dados embarcados open source. Além disso, como foram utilizadas tecnologias e recursos de software livre para a implementação do protótipo que serve como prova de conceito para a metodologia proposta, espera-se que a solução proposta seja adotada a fim de evitar o alto custo de licenciamento das soluções oferecidas no mercado atualmente.

7.2 Contribuições

Como principais contribuições originadas da pesquisa, enumera-se:

1. Definição de uma metodologia que contempla os requisitos necessários para o controle de acesso granular aos dados;
2. Aplicação do conceito de interceptação do comando SQL no método proposto;
3. Reescrita implícita e imperceptível em relação ao usuário final do banco de dados;

4. Controle de acesso que limita o acesso ao banco de dados apenas ao mecanismo de conexão criado obrigando, desta forma, que as políticas de controle de acesso granular sejam impostas;
5. Desenvolvimento, utilizando recursos de software livre, de um protótipo capaz de interceptar, analisar e reescrever os comandos SQL; e
6. Desenvolvimento, utilizando recursos de software livre, de uma interface gráfica acoplada ao mecanismo proposto para administrar e gerenciar as políticas de acesso.

7.3 Trabalhos Futuros

Conforme citado anteriormente, a linha de pesquisa deste trabalho trata de uma área antiga, porém ainda aberta para a resolução de desafios e que necessita de soluções para os mesmos. Desta forma, enumeraram-se alguns trabalhos futuros que podem se basear neste trabalho realizado:

1. Expandir o método proposto para sistemas de:
    a. Banco de Dados Embarcados;
    b. Banco de Dados Objeto-Relacional;
    c. Banco de Dados Orientados a Objetos; e
    d. Banco de Dados Relacionais licenciados;
2. Adequar a ferramenta proposta para:
    a. Outros SGBD open source utilizados no mercado, tais como PostgreSQL e Firebird;
    b. Trabalhar com outro mecanismo (*driver*) de conexão ao banco de dados (ODBC, por exemplo);
    c. Evitar a perda de desempenho por conta dos filtros com valores literais nos comandos SQL após a adição do predicado;
    d. Tratar comandos SQL complexos (Subconsultas aninhadas, por exemplo);
    e. Tratar políticas incompatíveis entre si; e
    f. Implementar o módulo Policy Forcer, para evitar que conexões oriundas de um mecanismo que não seja o Grão tenham acesso ao SGBD.

Referências Bibliográficas

ABNT NBR ISO/IEC 27002 – Tecnologia da Informação – Técnicas de Segurança – Código de prática para a gestão da segurança da informação, 2005.

Actian Corporation Website. Último Acesso: (12/2012). URL: http://www.actian.com/products/ingres

Adobe Flex Site Oficial. Último Acesso: (11/2012). URL: http://www.adobe.com/br/products/flex.html

Agrawal, R. Kiernan, J. Srikant, R. Xu, Y. Hippocratic databases. Proceedings of the 28th Conference on Very Large Databases, Hong Kong, China, 2002.

Agrawal, R. Bird, P. Grandison , T. Kiernan, J. Logan, S. Rjaibi, W. Extending Relational Database Systems to Automatically Enforce Privacy Policies, Proceedings of the 21st International Conference on Data Engineering, Tóquio, Japão, pág.1013-1022 , 2005.

Ausanka-Crues, R. Methods for Access Control: Advances and Limitations, pág 2, 2006.

Azevedo, L. Puntar, S. Thiago, R. Baião, F. Cappelli, C. Avaliação Prática de Funcionalidades para Autorização de Informações (Label Security e Virtual Private Database). Relatórios Técnicos do Departamento de Informática Aplicada da UNIRIO, 2010.

Ben-Natan, R. Implementing Database Security and Auditing, Elsevier Digital Press, 2005.

Bergh, J. RACF DB2 V8 Row Level Security - User Experiences, 2006. Último Acesso: (04/2012). URL: ftp://ftp.boulder.ibm.com/s390/zos/racf/pdf/r06_db2_v8_user_experience.pdf

Bertino, E. Samarati, P. Jajodia, S. An Extended Authorization Model for Relational Databases, IEEE Transactions on Knowledge and Data Engineering, v.9 n.1, pág.85-101, 1997.

Bertino, E. Sandhu, R. Database Security-Concepts, Approaches, and Challenges, IEEE Transactions on Dependable and Secure Computing, v.2 n.1, pág.2-19, 2005.

Bird, P. Implementing Low Level Access Control With DB2 UDB. The IDUG Solution Journal v.7 n.3,2000.

Blaze, M. Feigenbaum, J. Lacy, J. Decentralized Trust Management, Proceedings of the 1996 IEEE Symposium on Security and Privacy, pág.164, 1996.

BlazeDS Developer Guide. Último Acesso: (11/2012). URL: http://livedocs.adobe.com/blazeds/1/blazeds_devguide

Booch, G. Jacobson, I. Rumbaugh, J. OMG-Unified Modeling Language version 1.4, 2001.

Castano, S. Database Security. Addison-Wesley, 1995.

Chang, YC. Hill, M. Method and apparatus for query rewrite with auxiliary attributes in query processing operations. United States Patent 8122046, 2012.

Chaudhuri, S. Dutta, T. Sudarshan, S. Fine Grained Authorization Through Predicated Grants. Pág.1174-1183, 2007.

Contents of the Adaptive Server Enterprise 12.5 Collection. Último Acesso: (08/2012). URL: http://manuals.sybase.com/onlinebooks/group-as/asg1250e

Dastjerdi, A. Pieprzyk, J. e Naini, R. Security in databases: A survey study, 1996.

DB2 Database Software Website. Último Acesso: (08/2012). URL: http://www-01.ibm.com/software/data/db2/

Dodonov , E. Sousa, J. Guardia, H. Gridbox: securing hosts from malicious and greedy applications. In Proceedings of the 2nd workshop on Middleware for grid computing, pág 17-22, 2004.

Elmasri, R. E., Navathe, S. Sistemas de Banco de Dados. Addison-Wesley, 2005.

Ferraiolo, D. Kuhn, R. Role-based access control. In Proceedings of the NIST-NSA National (USA) Computer Security Conference, Baltimore, EUA, pág 554-563, 1992.

Ferraiolo, D. Cugini, J. Kuhn, R. Role-based access control: Features and motivations. In Proceedings of the Annual Computer Security Applications Conference, New Orleans, EUA, 1995.

Ferrari, E. Access Control in Data Management Systems. Synthesis Lectures on Data Management. Morgan and Claypool Publishers, 2010.

Feuerstein, S. Oracle PL/SQL Programming: Guide to Oracle8i Features. O'Reilly Media, 1999. Último Acesso. (08/2012)
URL: http://docstore.mik.ua/orelly/oracle/guide8i/ch08_01.htm

Fraga, R. Revista Java Magazine 72, 2009. Artigo: Integrando Flex com Java utilizando o BlazeDS.
Último Acesso: (11/2012).URL: http://www.devmedia.com.br/artigo-java-magazine-72-integrando-flex-com-java-utilizando-o-blazeds/14293

Garbus, J. Chang, A. Tyrrel, G. , Garbus, P. Administrator's Guide to Sybase ASE 12.5, Wordware Publishing, 2002.

Griffiths, P. Wade, B. “An Authorization Mechanism for a Relational Database System,” ACM TODS, vol. 1, no. 3, pág. 242–255, 1976.

Gollman, D. Computer Security. Wiley, 1999.

(GhostMouse, 2012) – GhostMouse Website. Último Acesso: (08/2012). URL: http://www.ghost-mouse.com/

H2 Database Engine Site Oficial. Último Acesso: (08/2012). URL: http://www.h2database.com/html/main.html

Halder, R., Cortesi, A. Observation-based fine grained access control for relational databases, pág 254-265, 2010.

Harris, S. CISSP Certification Exam Guide. McGraw-Hill/Osbourne, 2002.

Huwei, G. H.S. Ip, H. Yanchun, Z. Java-Based Approaches for Accessing Databases on the Internet and a JDBC-ODBC Implementation. Computing & Control Engineering Journal Vol. 9, No. 2, pág.71-78, 1998.

IBM Website. Último Acesso: (08/2012). URL: http://www.ibm.com

International Electrotechnical Commission – IEC Site Oficial. Último Acesso: (08/2012). URL: http://www.iec.ch

International Organization For Standardization – ISO Site Oficial. Último Acesso: (08/2012). URL: http://www.iso.org/iso/home.html

JAVA Site Oficial. Último Acesso: (11/2012). URL: http://www.java.com

JavaTM 2 SDK Standard Edition Documentation Version 1.4.2. Oracle Corporation. Último Acesso: (08/2012).
URL: http://docs.oracle.com/javase/1.4.2/docs/api/java/sql/PreparedStatement.html

Kabra, G. Ramamurthy, R. Sudarshan, S. Redundancy and information leakage in fine-grained access control, Proceedings of the 2006 ACM SIGMOD international conference on Management of data, Chicago, EUA, 2006.

Knox, D. Effective Oracle Database 10g Security by Design. McGraw-Hill. 2004.

Krutz, R. Vines, D. The CISSP and CAP Prep Guide, Platinum Edition. Indianapolis, Wiley Publishing Inc, 2007.

Lindblad, S. Security Without Multiple Databases: Oracle's Virtual Private Database. Oracle Open World 2001. Berlin, 2001.

Marques, M. Automatizando a Segurança de Dados utilizando Virtual Private Database (VPD), 2005.

Melton, J. (ISO-ANSI Working Draft) Database Language SQL (SQL3), ANSI TC X3H2, 1994.

Microsoft Website. Último Acesso: (07/2012). URL: http://www.microsoft.com

Microsoft SQL Server Site Oficial. Último Acesso: (08/2012). URL: http://www.microsoft.com/sqlserver/2008/pt/br/default.aspx

Microsoft SQL Server 2005 Website. Último Acesso: (08/2012). URL: http://www.microsoft.com/brasil/servidores/sql/2005/default.mspx

Moraes, M. Vasconcelos, A. ArcADe: Um Modelo de Processo para Análise e Projeto Baseado em Arquitetura de Software, 2004.

Mouelhi, T. Fleurey, F. Baudry, B. Le Traon, Y. Mutating DAC And MAC Security Policies : A Generic Metamodel Based Approach, pág 5, 2008.

MySQL: the world's most popular open source database. Último Acesso: (08/2012). URL: http://www.mysql.com

Negrello, F. Wainer, J. Revogação com Downgrade, 2006.

Olson, L. Gunter, C. Madhusudan, P. A formal framework for reflective database access control policies, Proceedings of the 15th ACM conference on Computer and communications security, Alexandria, EUA, 2008.

Opyrchal, L. Cooper, J. Poyar, R. Lenahan, B. Zeinner, D. Bouncer: Policy-Based Fine Grained Access Control in Large Databases. International Journal of Security and Its Applications, 2011.

Oracle8i Database Online Documentation. Último Acesso: (08/2012). URL: http://tahiti.oracle.com/pls/tahiti/tahiti.homepage

Oracle Website. Último Acesso: (08/2012). URL: http://www.oracle.com/index.html

Oracle Database Java Developer's Guide. Último Acesso: site Stanford University (08/2012). URL: http://www.stanford.edu/dept/itss/docs/oracle/10g/java.101/b12021/storproc.htm

Oracle Database Licensing Information. Último Acesso: site Stanford University (08/2012). URL: http://www.stanford.edu/dept/itss/docs/oracle/10g/license.101/b13552/editions.htm

Oracle Database PL/SQL Packages and Types Reference 11g Release 1.Último Acesso: site Oracle Documentation (08/2012). URL: http://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_rls.htm

Oracle Database Security Guide.
Último Acesso: site Oracle Documentation (08/2012). URL: http://download.oracle.com/docs/cd/B28359_01/network.111/b28531/vpd.htm

Oracle Label Security Administrator's Guide. Último Acesso: site Oracle Documentation (08/2012). URL: http://download.oracle.com/docs/cd/E18283_01/network.112/e10745/intro.htm#i1006974

Oracle Label Security with Oracle Database 11g Release 2, 2009. Último Acesso: site Oracle Documentation (08/2012). URL: http://www.oracle.com/us/products/financing/owp-security-label-security-11gr2-133601.pdf

Oracle Solaris 11 – Oracle Website.
Último Acesso: (08/2012). URL: http://www.oracle.com/us/products/servers-storage/solaris/solaris11/overview/index.html

Oracle Technology Global Price List, 2012. Último Acesso: site Oracle Technology (08/2012). URL: http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

Patel, P. Moss, K. Java Database Programming with JDBC, 2nd Edition, Coriolis Group Books, 1997.

PostgreSQL: the world's most advanced open source database. Último Acesso: (08/2012). URL: http://www.postgresql.org

Purevjii, B. Aritsugi, M. Imai, S Kanamori, Y. An Implementation Design of a Fine-Grained Database Access Control Policy Consistency Checking Mechanism, Knowledge-Based Intelligent Information and Engineering Systems and the XVII Italian Workshop on Neural Networks on Proceedings of the 11th International Conference, Vietri sul Mare, Itália, 2007.

Rabitti, F. Bertino, E. Kim, W. Woelk, D. A model of authorization for next-generation database systems, ACM Transactions on Database Systems (TODS), v.16 n.1, pág .88-131, 1991.

Rayns, C. Grant, R. Holmans, M. Kappeler, P. Ogata, J. Verbestel, N. Yates, M. Multilevel Security and DB2 Row-Level Security Revealed, 2005. Último Acesso: Site Redbooks (08/2012).
URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg246480.pdf.

Reese, G. Database Programming with JDBC and Java, 2nd Edition, O'Reilly, 2000.

Resource Access Control Facility. Último Acesso: (08/2012). URL: http://www-03.ibm.com/systems/z/os/zos/features/racf

Ritchie, D. The Development of the C Language. Último Acesso: site Bell Laboratories (04/2012). URL: http://cm.bell-labs.com/cm/cs/who/dmr/chist.html

Rizvi, S. Mendelzon, A. Sudarshan, S. Roy, P. Extending Query Rewriting Techniques for Fine-Grained Access Control, 2004.

Rushby, J. The Bell and La Padula security model. Draft report, Computer Science Laboratory, Pág 6, 1984.

SAP Website. Último Acesso: (08/2012). URL: http://www.sap.com/brazil/index.epx

Sasaoka, L. K. Controle de Acesso em Bancos de Dados Geográficos. Dissertação de Mestrado, Universidade Estadual de Campinas, 2002.

Security-Enhanced Linux (SELinux). Acesso: (08/2012). URL: http://www.nsa.gov/research/selinux/index.shtml.

Shaw, M. Garlan. D. Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, New Jersey, 1996.

Shi, J. Zhu, H. A fine-grained access control model for relational databases. Journal of Zhejiang University - Science C, 11(8):575--586, 2010.

SQL Server White Paper – Do Not Pay Too Much for Your Database Licenses, 2010. Último Acesso: (08/2012). URL: http://database5.com/d/do-not-pay-too-much-for-your-database-licenses-w2414.html

SQuirreL SQL Client. Último Acesso: (11/2012). URL: http://squirrel-sql.sourceforge.net/#overview

Stone, J. Merrion, S. Instant Messaging or Instant Headache? ,2004. Último Acesso. (08/2012). URL: http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=142

Stonebraker, M. Wong, E. Access control in a relational data base management system by query modification, Proceedings of the 1974 ACM annual conference, pág 180-186, San Diego, EUA, 1974.

Sybase Website. Último Acesso: (08/2012). URL: http://www.sybase.com

Sybase Technical Library – System Administration Guide (Online Only). Último Acesso: (04/2012). URL: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/sag/@Generic__BookTextView/38430;hf=0

Tešanovi´c, A. Nyström, D. Hansson, J. Norström, C. Embedded Databases for Embedded Real-Time Systems: A Component-Based Approach. Technical Report MRTC Report ISSN 1404-3041 ISRN MDH-MRTC-43/2002-1-SE, Dept. of Computer Engineering, M¨alardalen University, 2002.

The Java Tutorials. Oracle CORPORATION. Último Acesso:(08/2012). URL: http://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

Vargas, T. A história de UML e seus diagramas, 2008.

Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, S. Samarati, P. Assessing Query Privileges via Safe and Efficient Permission Composition, 2008.

Vinod, A. A Database Level Implementation to Enforce Fine Grained Access Control, pág 15, 2008.

Wang, Q. Yu, T. Li, N. Lobo, J. Bertino, E. Irwin, K. Byun, J.-W. On the correctness criteria of fine-grained access control in relational databases, 2007.

Wei, K Muthuprasanna, M. Kothari, S. Eliminating SQL Injection Attacks in Stored Procedures, pp. 191-198, IEEE ASWEC, 2006.

Anexo A

O Anexo A apresenta um exemplo de utilização do VPD (Virtual Private Database) da Oracle. O exemplo consiste na criação de uma política de segurança simples criada por Knox (2004), a qual consiste em limitar o acesso aos dados da tabela de cadastro de funcionários chamada EMP (descrita no Código A.1):

Código A.1. Descritivo da tabela EMP.

```
describe emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

A coluna DEPTNO, armazena informações do departamento em que o funcionário está lotado. O objetivo da política é permitir que apenas os dados do departamento 10 não sejam visíveis. Para este fim, pode ser criada uma função PL/SQL conforme mostrado no Código A.2.

Código A.2. Criação da função no_dept10.

```
CREATE OR REPLACE FUNCTION no_dept10 (
  2    p_schema  IN  VARCHAR2,
  3    p_object  IN  VARCHAR2)
  4    RETURN VARCHAR2
  5  AS
  6  BEGIN
  7    RETURN 'deptno != 10';
  8  END;
  9  /

Function created.
```

Apenas a criação da função acima não é o suficiente para limitar o acesso aos dados. Além dela, para este exemplo, deve-se associar a função no_dept10 à tabela EMP. Isto deve ser feito através do pacote DBMS_RLS, conforme descrito no Código A.3, onde é criada uma política chamada *quickstart*, associando a função no_dept10 à tabela EMP.

Código A.3. Adição de política utilizando o pacote dbms_rls.

```
BEGIN
  2    DBMS_RLS.add_policy
  3                (object_schema      => 'SCOTT',
  4                 object_name        => 'EMP',
  5                 policy_name        => 'quickstart',
  6                 policy_function    => 'no_dept10');
  7  END;
  8  /

PL/SQL procedure successfully completed.
```

Para testar o funcionamento da política criada, é necessário executar alguma consulta ou comando de manipulação de dados para confirmar o funcionamento da referida política. O comando SELECT no Código A.4 permite verificar que o departamento 10 não foi listado no resultado da consulta.

Código A.4. Consulta para testar o funcionamento da política.

```
SELECT DISTINCT deptno FROM emp;

    DEPTNO
----------
        20
        30
```

Na verdade, de maneira implícita foi adicionado um predicado à cláusula WHERE do comando do Código A.4. Portanto, o referido comando SELECT, na realidade, foi modificado de acordo com o Código A.5.:

Código A.5. Consulta alterada através do mecanismo VPD.

```
SELECT DISTINCT deptno FROM emp
WHERE deptno != 10;
```

Anexo B

O Anexo B apresenta um exemplo de utilização do Row-Level Authorization do Sybase descrito por Garbus et al (2002). O exemplo consiste na criação de uma política de segurança para acesso à tabela chamada TESTTAB1. Tal tabela será utilizada para armazenar dados de funcionários. As colunas usadas como filtro serão o código do departamento (DEPTID), código do funcionário (EMPID), nome e telefone do funcionário (NAME, PHONE).

Passo 1 – Criação das regras de acesso empid1_access e deptno1_access, mostrado no Código B.1.

Código B.1. Criação de regras de acesso no Sybase.

```
create access rule empid1_access  as @empid = 1
create access rule deptno1_access as @deptid = 2
```

Passo 2 – Criação das regras de acesso do tipo "OU" (operador lógico OR) name1_access e phone_access, mostrado no Código B.2.

Código B.2. Criação de regras de acesso no Sybase – operador lógico OR.

```
create or access rule name1_access as @name = "smith"
create or access rule phone_access  as @phone ="9999"
```

Passo 3 – Criação da tabela onde o controle de acesso granular irá atuar, mostrado no Código B.3.

Código B.3. Criação de tabela testtab1 no Sybase para posterior controle de acesso granular.

```
create table testtab1 (empno int, deptno int,name char(10),
phone char(4))
```

Passo 4 – Associação das regras de acesso às referidas colunas da tabela testtab1, mostrado no Código B.4.

Código B.4. Associação das colunas da tabela testtab1 com as políticas de acesso criadas.

```
sp_bindrule empid1_access  ,"testtab1.empno"
sp_bindrule deptno1_access ,"testtab1.deptno"
sp_bindrule name1_access   ,"testtab1.name"
sp_bindrule phone_access   ,"testtab1.phone"
```

Passo 5 – Inclusão de registros na tabela testtab1, mostrado no Código B.5.

Código B.5. Inclusão de registros na tabela testtab1 para verificar a atuação das políticas de acesso criadas.

```
insert testtab1 values(1,1,"smith","3245")
insert testtab1 values(2,1,"jones","0283")
insert testtab1 values(1,2,"smith","8282")
insert testtab1 values(2,2,"smith","9999")
insert testtab1 values(3,2,"smith","8888")
insert testtab1 values(1,2,"jones","9999")
insert testtab1 values(2,3,"jones","9999")
(1 row affected)
```

Passo 6 - Verificar o resultado das regras de acesso criadas através de comandos SQL SELECT, mostrado no Código B.6.

Código B.6. Comando SQL alterado através das políticas de acesso criadas.

```
/** As regras de acesso criaram o seguinte predicado:
*   WHERE empno = 1 and deptno = 2
*     and ( name = "smith" or phone = "9999" ) */
select * from testtab1

empno deptno name phone
----- ------ ----- -----
1          2 smith 8282
1          2 jones 9999
```

Passo 7 – Desvincular a regra de acesso associada a coluna empno e verificar o resultado, mostrado no Código B.7.

Código B.7. Comando para desvincular a coluna testtab1.empno das regras de acesso.

```
sp_unbindrule "testtab1.empno",NULL,"accessrule"

/** Após desvincular regra de acesso à coluna empno será criado o
*   seguinte predicado:
*   WHERE deptno = 2 and ( name = "smith" or phone = "9999" )
*/

select * from testtab1

empno deptno name phone
----- ------ ----- -----
1          2 smith 8282
2          2 smith 9999
3          2 smith 8888
1          2 jones 9999
(4 rows affected)
```

Anexo C

O Anexo C apresenta um exemplo de utilização do Multiple Level Security (DB2) do DB2. O exemplo consiste em controlar o acesso aos dados da tabela de preços por loja da empresa fictícia listada abaixo. A tabela possui quatro colunas: Loja, Produto, Descrição e Preço, conforme mostrada na Figura C.1.

Figura C.1: Exemplo de tabela par armazenamento de preços por loja da organização fictícia.

Store	Inventory #	Description	Price
AZS1	ABC	SHIRT	11.95
AZS2	ABC	SHIRT	11.95
AZS1	DEF	PANT	21.95
AZS2	DEF	PANT	21.95
CAS2	ABC	SHIRT	18.95
CAS1	DEF	PANT	28.95
CAS2	DEF	PANT	28.95
WAS1	ABC	SHIRT	12.95
WAS1	GHI	SHOE	31.95
WAS2	ABC	SHIRT	12.95
COS1	ABC	SHIRT	13.95
COS2	DEF	PANT	23.95
COS2	GHI	SHOE	32.95
MNS1	ABC	SHIRT	14.95
MNS2	ABC	SHIRT	14.95
MNS1	DEF	PANT	24.95
MNS2	GHI	SHOE	33.85

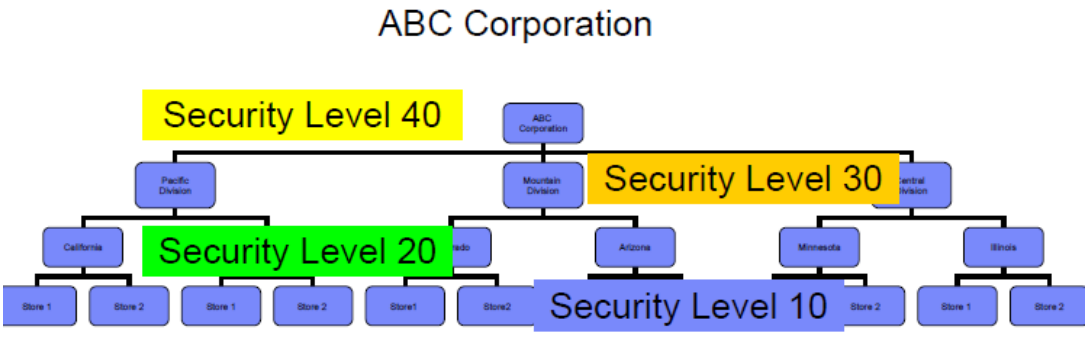
Fonte: Bergh,2006.

O objetivo é definir quatro níveis de segurança:

- Nível de Corporação (Nível de Segurança 40 – Amarelo);
- Nível de Divisão (Nível se Segurança 30 – Laranja);
- Nível de Estado (Nível de Segurança 20 – Verde); e
- Nível de Loja (Nível de Segurança 10 – Azul).

A Figura C.2 ilustra os níveis de segurança propostos.

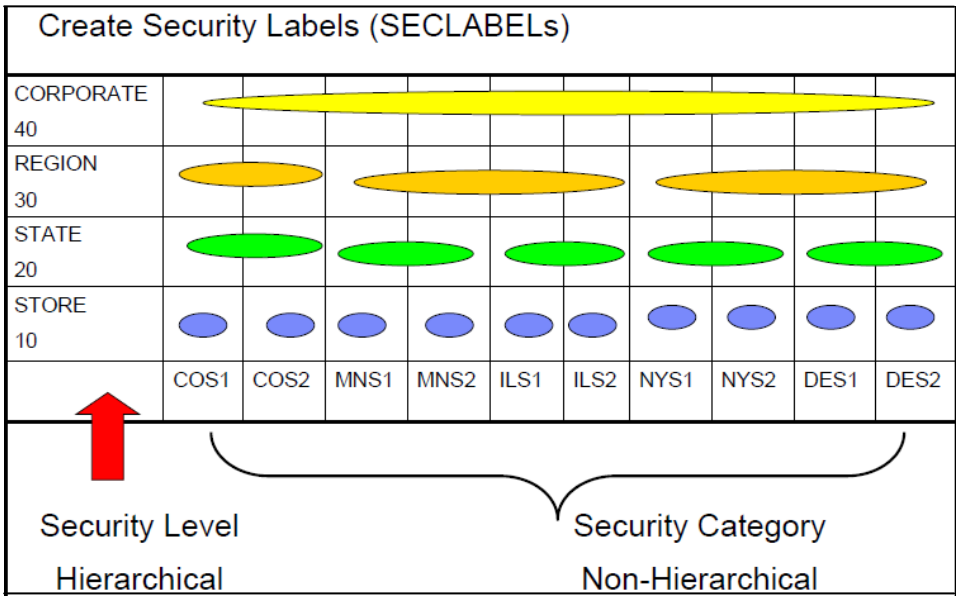
Figura C.2: Níveis de segurança definidos para a organização ABC



Fonte: Bergh, 2006.

Da necessidade de controlar o acesso aos dados a partir dos níveis de segurança listados anteriormente, é possível definir os rótulos de segurança conforme Figura C.3.

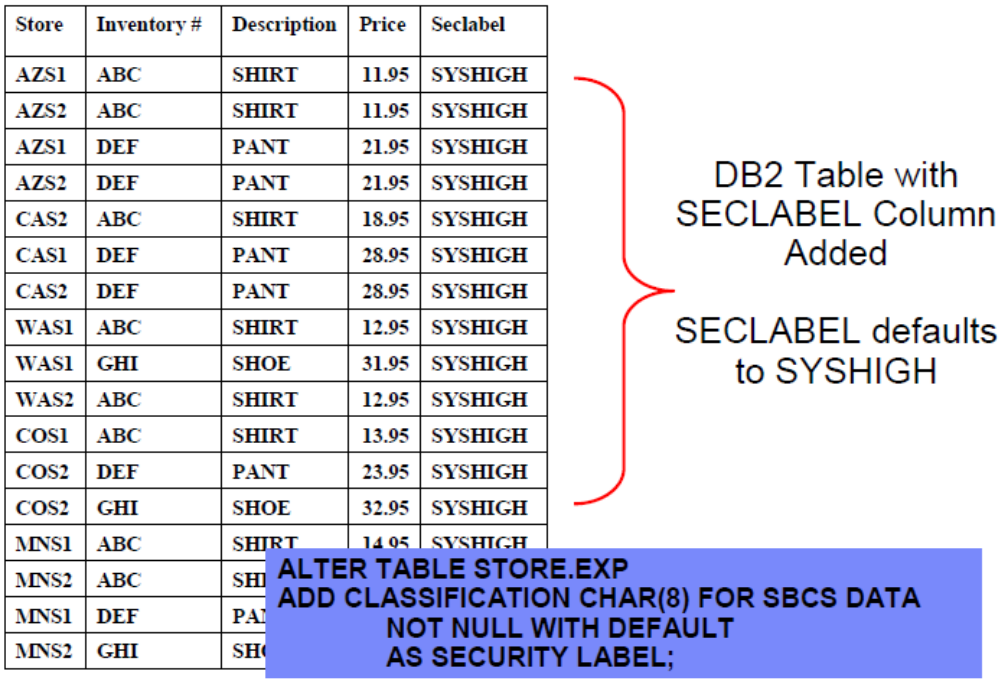
Figura C.3: Exemplo de criação dos rótulos de segurança para a organização ABC.



Fonte: Bergh, 2006.

Definidos os rótulos de segurança (SECLABELs), deve-se adicionar uma coluna à tabela de Preços para armazenar o SECLABEL. O valor padrão para a coluna adicionada será SYSHIGH, o que indica que somente os administradores terão acesso aos dados da tabela enquanto não forem definidos os rótulos de segurança para acesso, como mostrado na Figura C.4.

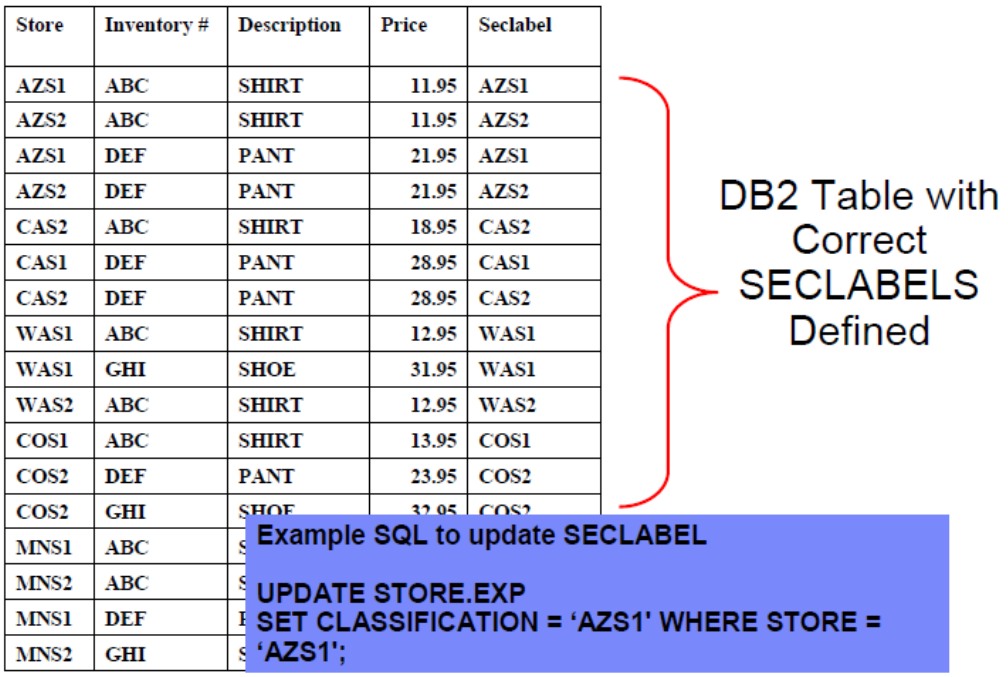
Figura C.4: Exemplo de alteração do modelo de dados para criação de coluna SECLABEL



Fonte: Bergh, 2006.

Depois de adicionada a coluna, é necessário atualizar a coluna SecLabel com os rótulos de segurança preenchidos, como mostrado na Figura C.5:

Figura C.5: Exemplo de atualização da coluna SECLABEL com os rótulos corretos



Fonte: Bergh, 2006.

No entanto, o trabalho não termina neste ponto. Ainda será necessário adicionar as regras de controle do recurso no RACF. Este processo, por não fazer parte do escopo do estudo, não será detalhado.

Anexo D

O Anexo D apresenta a funcionalidade de cada uma das entidades e atributos exibidos na Figura 6.2.

- **Perfil** – Cadastro dos perfis de acesso aos dados.

Quadro D.1: Lista de atributos da entidade Perfil.

Atributo	Descrição
codigo	Código do Perfil
descricao	Descrição do Perfil

- **Usuario** – Cadastro dos usuários que serão gerenciados pelas políticas de segurança.

Quadro D.2: Lista de atributos da entidade Usuario.

Nome do Atributo	Descrição
usuarioBanco	*Login* do usuário que se conectará ao banco de dados.
descricao	Descrição / Nome do usuário

- **Objeto** – Cadastro dos Objetos.

Quadro D.3: Lista de atributos da entidade Objeto.

Nome do Atributo	Descrição
codigo	Código do Objeto
nome	Nome do Objeto
descricao	Descrição do Objeto
tipo	Tipo do Objeto (Tabela ou Visão)
nomeEsquema	Nome do esquema onde o objeto reside.

- **Atributo** – Cadastro dos atributos de cada Objeto.

Quadro D.4: Lista de atributos da entidade Atributo.

Nome do Atributo	Descrição
codigo	Código do Atributo
nome	Nome do Aributo
tipoDado	Tipo de dados do atributo (String, Numérico ou Data)

- **Relacionamento** – Cadastro dos relacionamentos entre os Objetos.

Quadro D.5: Lista de atributos da entidade Relacionamento.

Nome do Atributo	Descrição
codigo	Código do Relacionamento
atributo	Nome do Atributo que está sendo relacionado
objetoRelacionado	Código do Objeto de Relacionamento
atributoRelacionado	Nome do Atributo de Relacionamento

- **Política** – Cadastro das Políticas de Segurança.

Quadro D.6: Lista de atributos da entidade Política.

Nome do Atributo	Descrição
Código	Código da Política de Segurança
nome	Nome da Política de Segurança
descricao	Descrição da Política de Segurança
ativa	Indicador se a política está ativa (Sim/Não)
disparaEmSelect	Indicador se a política dispara em comandos SQL Select
disparaEmInsert	Indicador se a política dispara em comandos SQL Insert
disparaEmUpdate	Indicador se a política dispara em comandos SQL Update
disparaEmDelete	Indicador se a política dispara em comandos SQL Delete

- **PolíticaObjeto** – Associação das Políticas de Segurança com os Objetos.

Quadro D.7: Lista de atributos da entidade PolíticaObjeto.

Nome do Atributo	Descrição
codigo	Código da Associação entre a Política e o Objeto
condicaoAcesso	Condição de Acesso da Política para o Objeto. Os valores possíveis são "Tudo", "Nada" ou "Através de Atributos".
tipoObjetoPolitica	Participação do objeto na Política ("Principal" ou "*Lookup*").

- **PolíticaAtributo** – Associação dos Objetos das Políticas de Segurança com os Aributos.

Quadro D.8: Lista de atributos da entidade PolíticaAtributo.

Nome do Atributo	Descrição
codigo	Código da Associação entre os Objetos da Política e os Atributos.
acessível	Indica se o atributo é acessível no Select.
operadorComparacao	Indica o operador de comparação que será utilizado como filtro para o atributo. Os valores possíveis são “Maior que”, “Menor que”, “Maior ou Igual a”, “Menor ou igual a” “Entre”, “Igual”, “Diferente”.
valor	Indica o Valor que o aributo respeitará em relação ao operador Comparação.
valorLimite	Indica o Valor Limite quando o operadorComparacao for o “Entre”.
operadorLogico	Indica o operador lógico utilizado com o próximo filtro. Os valores possíveis são “And”, “Or” e “Not”.