UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n°0002/2010

Avaliação Prática de Funcionalidades para Autorização de Informações (Label Security e Virtual Private Database)

Leonardo Guerreiro Azevedo
Sergio Puntar
Raphael Melo Thiago
Fernanda Baião
Claudia Cappelli

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Projeto de Pesquisa

Grupo de Pesquisa Participante
NP2Tec

Patrocínio
PETROBRAS

Relatórios Técnicos do DIA/UNIRIO, No. 0002/2010
Editor: Prof. Sean W. M. Siqueira Fevereiro, 2010

Avaliação Prática de Funcionalidades para Autorização de Informações (Label Security e Virtual Private Database)*

Leonardo Guerreiro Azevedo, Sergio Puntar, Raphael Melo Thiago, Fernanda Baião, Claudia Cappelli

Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

{azevedo, sergio.puntar, raphael.thiago, fernanda.baiao, claudia.cappelli}@uniriotec.br

Abstract. Information security is a hot topic for commercial and government organizations. The implementation of information security has the goal to guarantee that only authorized users access data. There are different mechanisms for information authorization. This work evaluates the implementations of RBAC and MAC in Oracle DBMS, named as VPD and Label Security, respectively. Besides, a flexible model for authorization rules definition is presented. This model is evaluated, related to VPD, based on model, data and queries of TPC-H benchmark.

Keywords: RBAC (Role-Based Access Control), MAC (Mandatory Access Control), VPD (Virtual Private Database), Label Security, Benchmark TPC-H, Flexible Model for Information Authorization.

Resumo. Segurança da informação é um tema essencial em organizações comerciais e governamentais, e sua operacionalização requer a existência de funcionalidades que permitam garantir que apenas as pessoas devidamente autorizadas acessem os dados. Existem diferentes mecanismos para tratar autorização de acesso. Este trabalho apresenta a avaliação das implementações dos mecanismos RBAC e MAC no SGBD Oracle, VPD e Label Security respectivamente. Além disso, um modelo flexível para definição das regras de autorização é apresentado. Este modelo foi avaliado, em relação ao VPD com base no modelo, dados e consultas do benchmark TPC-H.

Palavras-chave: RBAC (Role-Based Access Control), MAC (Mandatory Access Control), VPD (Virtual Private Database), Label Security, Benchmark TPC-H, Modelo flexível para autorização de informações.

---

* Trabalho patrocinado pela Petrobras.

Sumário

Figuras

1 Introdução

A pesquisa em segurança da informação tem recebido cada vez mais atenção a fim de atender as necessidades de segurança de aplicações comerciais e governamentais. A integridade, disponibilidade, e confidencialidade de sistemas de software, bancos de dados e redes de dados são as principais preocupações de todos os setores das organizações. O acesso e revelação não autorizada de recursos corporativos podem impactar fortemente as operações das organizações e levar a sérios problemas financeiros, legais, de segurança pessoal, privacidade e confidencialidade. As maiores questões de segurança e confidencialidade se encontram na integridade da informação, a qual é garantida por mecanismos de controle (ou autorização) de acesso [Sandhu *et al.* 1996]. Estes mecanismos fazem com que regras de negócio do tipo assertiva de ação de autorização sejam garantidas.

Segundo [BRG 2009], regra de negócio é uma declaração que define ou restringe algum aspecto de uma organização. Regras de negócio têm como objetivo definir a estrutura de um negócio ou controlar ou influenciar o seu comportamento. Em particular, uma categoria de regras de negócio, definida pela BRG [2009], é a regra de assertiva de ação de autorização, ou *regra de autorização*, a qual restringe **quem** é permitido realizar uma **ação** na organização sobre quais **informações**.

Dentre os mecanismos existentes para autorização de acesso, [Chen e Crampton 2009] apontam que Role-Based Access Control (RBAC), proposto inicialmente por [Ferraiolo e Khun, 1992], tem recebido foco principal nas pesquisas recentes e são amplamente aceitos como uma alternativa aos mecanismos tradicionais, tais como DAC (Discricionary Access Control) e MAC (Mandatory Access Control), inicialmente propostos por [DoD 1983].

O mecanismo DAC restringe acesso a objetos baseado na identidade de usuários e/ou grupos aos quais eles pertencem [Ferraiolo e Khun, 1992]. [Yang 2009] aponta que políticas DAC não garantem controle sobre o fluxo de informações; dessa forma, torna possível que informações vazem para usuários que não têm permissão de lê-las.

Por outro lado, [Yang 2009] caracteriza as políticas MAC, também conhecidas como *label security*, como sendo baseadas em regulamentações mandatórias determinadas por uma autoridade central. A forma mais comum de MAC é a política de segurança de múltiplos níveis usando classificação de sujeitos (usuários ou grupos) e objetos (dados) dos sistemas. MAC controla o fluxo de informações. No entanto, MAC não aborda as ações que podem ser executadas pelos sujeitos sobre os dados [Ferraiolo e Khun, 1992]. Além disso, como um rótulo é atribuído a cada instância de dados, se existirem muitas políticas definidas, o custo de gestão e, principalmente, manutenção dos rótulos é alto, enquanto que a flexibilidade (facilidade de manutenção dos valores dos rótulos) é baixa.

No caso de RBAC, o controle de acesso considera funções e informações, ao invés de estritamente informações. Neste caso, o interesse principal é proteger a integridade da informação: **quem** pode realizar qual **ação** sobre qual **informação** [Ferraiolo e Khun, 1992]. Em [Ferraiolo *et al.* 2001] é proposto um padrão para RBAC a fim de unificar idéias existentes em diferentes modelos de referência de RBAC, produtos comerciais e protótipos de pesquisa.

Os mecanismos para controle de acesso aparecem implementados em diferentes Sistemas Gerenciadores de Banco de Dados (SGBD), como, por exemplo, Oracle, Sybase, Microsoft SQL-Server [Yang 2009]. Este relatório tem o objetivo de avaliar as

funcionalidades do SGBD Oracle, em sua versão 10.2.0.4, para controle de autorização. As funcionalidades MAC (denominada *label security*, no caso do Oracle) e RBAC (denominada *Virtual Private Database*, no caso do Oracle) serão avaliadas. Esta avaliação é realizada através da análise de exemplos de regras de autorização baseados no modelo de dados e consultas do benchmark TPC-H [TPC-H, 2009], o qual foi utilizado para realização dos testes experimentais. Como contribuição consequente desta avaliação, um modelo de dados foi elaborado a fim de armazenar as regras de autorização de uma forma mais genérica.

Este relatório foi produzido pelo Projeto de Pesquisa em Autorização de Informação como parte das iniciativas dentro do contexto do Projeto de Pesquisa do Termo de Cooperação entre UNIRIO/NP2Tec e a PETROBRAS/TIC-E&P/GDIEP.

Esse relatório está organizado em 9 capítulos, sendo o capítulo 1 a presente introdução. No capítulo 2, são caracterizadas as funcionalidades de controle de acesso VPD (Virtual Private Database) e Label Security do Oracle. Neste capítulo, as funcionalidades são descritas e são apresentados exemplos de uso das mesmas. No capítulo 3, é apresentado o benchmark utilizado para realizar a avaliação das funcionalidades do Oracle. No capítulo 4, são apresentados exemplos de regras de autorização elaborados a partir dos exemplos levantados, mas considerando o TPC-H. No capítuloo 5, é apresentada a avaliação do Label Security para tratar as regras propostas, bem como para tratar escrita e trigger. No capítulo 6, é apresentada a proposta de modelo para armazenamento das regras de autorização utilizado para generalizar o uso do VPD, bem como as funções necessárias para execução das regras de autorização. No capítulo 7, são apresentadas as avaliações experimentais do modelo proposto e das funções relacionadas. Nos capítulo 8 e 9, são apresentadas as conclusões e referências do trabalho realizado, respectivamente.

2 Funcionalidades de controle de acesso

Esta seção analisa as funcionalidades VPD (Virtual Private Database) e Label Security disponíveis no SGBD Oracle. Para cada funcionalidade analisada será apresentada uma breve descrição, exemplos de uso, e uma análise da aplicabilidade aos requisitos levantados pelo projeto.

2.1 Virtual Private Database

2.1.1 Descrição da funcionalidade

O Virtual Private Database (VPD) permite reforçar a segurança diretamente em tabelas, visões e sinônimos, anexando diretamente nestes elementos as políticas de segurança, que são automaticamente ativadas sempre que um usuário tenta acessar dados. Como a política de segurança é aplicada à própria base, o uso de diferentes aplicações não irá contornar a segurança.

Quando o usuário tenta acessar (direta ou indiretamente) uma tabela, visão, ou sinônimo protegido por uma política do VPD, o servidor altera dinamicamente o comando SQL do usuário. Essa alteração cria uma condição WHERE (conhecida como predicado) retornada pela função que implementa a política de segurança. As políticas podem ser definidas para os comandos SELECT, INSERT, UPDATE, INDEX, e DELETE.

As funções que retornam os predicados podem chamar outras funções e com o pacote PL/SQL, é possível incluir chamadas em C ou em Java. Uma função de política de segurança pode retornar diferentes predicados para cada usuário, grupo de usuários ou aplicação. O contexto de aplicação permite acesso seguro aos atributos utilizados como base para as políticas. Por exemplo, o usuário com o atributo de posição "Gerente" possui uma política de segurança diferente de um usuário com o atributo de posição "Empregado".

O VPD pode ainda ser aplicado a nível de coluna, nesse caso somente em tabelas e visões. Ao especificar uma coluna que necessita de segurança, uma política será aplicada sempre que a coluna for referenciada. Por exemplo, um usuário faz a seguinte consulta em uma tabela emp:

SELECT nome, mat FROM emp;

Contudo, a tabela emp possui uma política protegendo a coluna mat, onde cada usuário pode visualizar apenas a sua própria matrícula, então a função retorna o seguinte predicado mat='minha_mat' e o comando é reescrito da seguinte forma:

SELECT nome, matricula FROM emp WHERE mat='minha_mat';

Além disso, se o que se deseja é apenas proteger o conteúdo específico da coluna, é possível mascará-la e, dessa forma, a consulta retornaria todos os valores da coluna mat como NULL.

Logo, as políticas criadas no VPD se encontram nos seguintes casos:

- **Regra de autorização sobre tabela**: Neste caso é necessário definir sobre qual tabela a restrição será aplicada. Toda vez que esta tabela aparecer na cláusula FROM de uma operação SQL ou DML (por default ela é aplicada para todas as consultas, SELECT, DELETE, UPDATE, INSERT), que tenha o acesso controlado, a regra será aplicada.
- **Regra de autorização sobre colunas**: Neste caso é necessário definir sobre qual tabela e colunas da tabela a regra de autorização deve ser aplicada. Isto é feito no momento da associação da regra de autorização à tabela (quando as colunas também são definidas). Por padrão, a regra de autorização é aplicada a todas as colunas. Ou seja, caso não seja informada nenhuma coluna, a regra é aplicada a qualquer operação executada sobre a tabela. Este é o caso apresentado no item anterior – regra de autorização sobre tabela.

  Portanto, no **caso** da regra de autorização sobre colunas, a autorização será aplicada quando as colunas com autorização controlada estiverem envolvidas na operação. Neste caso, serão retornados apenas os registros em que a regra de autorização retornar verdadeiro.

  Por outro lado, se estas colunas não forem utilizadas na operação, mesmo estando em uma tabela com a regra de autorização definida, o usuário terá acesso aos dados da tabela.
- **Regra de autorização para mascarar colunas específicas**: Este caso é semelhante ao caso anterior (Regra de autorização sobre colunas). A única diferença está no fato de que quando a regra de autorização é aplicada são retornadas todas as linhas que a consulta sem a aplicação da regra de autorização retornaria, porém as informações que o usuário não tem acesso são mascaradas.

2.1.2 Uso da funcionalidade

Essa seção apresenta exemplos práticos do uso do VPD, encontrados nos tutoriais que compõem a OBE (Oracle By Example) [Oracle, 2009]. No cenário a seguir serão utilizadas duas tabelas: EMPLOYEES e CUSTOMERS, além disso, é importante saber que o usuário utilizado para conectar ao banco de dados é o e-mail de cada um (que também está armazenado na tabela EMPLOYEES) e que a tabela CUSTOMERS armazena o nome dos clientes, o limite de crédito de cada um e o id do funcionário que gerencia a sua conta.

2.1.2.1 Visualizando os dados sem restrição

Primeiramente vamos ver o que acontece quando não existe nenhuma política de acesso aos dados. A Figura 1 apresenta um exemplo de consulta e a Figura 2 apresenta o resultado de execução desta consulta sem controlar o acesso.

```
connect "ELENI.ZLOTKEY@OSRD.COM"/welcome1
select cust_last_name, cust_first_name, credit_limit, account_mgr_id
 from oe.customers
 order by account_mgr_id;
```

Figura 1 – Exemplo de consulta para a qual se deseja controlar o acesso

```
Stockwell            Cary                 2300          149
Olin                 Cary                 2300          149
Ganesan              Clara                2300          149
Andrews              Ajay                 2300          149
Prashant             Kathy                2400          149
Neeson               Graham               2400          149

319 rows selected.
```

Figura 2 – Exemplo de resultado da consulta sem controle de acesso

Observe que nesse caso a funcionária Eleni, identificada por "ELENI.ZLOTECKEY@OSRD.COM" (Figura 1), pode visualizar o limite de crédito de todos os clientes (quarta coluna do resultado da consulta apresentado na Figura 2).

2.1.2.2 Aplicando políticas VPD em colunas relevantes

Agora vamos criar uma política VPD que restringe o acesso aos dados dos clientes somente aos seus respectivos gerentes. Como na tabela CUSTOMERS, o campo *account_mgr_id* identifica o *id* do gerente do cliente, o que precisamos fazer é criar uma política que quando um usuário consulta o salário do cliente, ele veja apenas as linhas em que o seu *id* aparece na coluna *account_mgr_id*.

O primeiro passo é criar uma função para essa política como, por exemplo, a que está representada na Figura 3. Os comentários em verde explicam detalhes da função, a qual recupera o usuário da sessão, em seguida recupera o *id* do empregado de acordo com o usuário da sessão e constrói o qualificador a ser adicionado na cláusula *where* da consulta realizada (na variável *out_string*).

Observe que, na função apresentada Figura 3, a variável *out_string* foi inicializada com valor *1=2*. Isto significa que, se o usuário não tiver acesso aos dados, a função retornará a *string 1=2*, a qual será concatenada na consulta com AND, resultando em AND 1=2. Dessa forma, nenhum dado será retornado. Caso contrário, o usuário tem acesso aos dados, e ao valor *default* será concatenada a string que faz a restrição

(representado pelo comando *out_string := out_string || 'or account_mgr_id = ' || v_employee_id;*), dando acesso ao usuário. O resultado obtido é *"1 = 2 or account_mgr_id = <id do empregado*).

```
create or replace function f_policy_customers
-- A função deve conter os seguintes parâmetros
   (schema in varchar2, tab in varchar2)
-- A função vai retornar uma string que será usada
-- como cláusula WHERE
 return varchar2
as
 v_employee_id number;
 v_user          varchar2(100);
 -- out_string será o retorno da função.
 -- É inicializada com '1=2' porque 'WHERE 1=2'
 -- significa que nada será acessado e, mais tarde, pode
 -- ser combinado com outras condições através do OR
 out_string      varchar2(400) default '1=2 ';
begin
  -- Capturar a sessão do usuário
  v_user := lower(sys_context('userenv','session_user'));
  -- Se a sessão não for SECUSER,
  --  então a conexão está usando um usuário personalizado do banco
  -- Se a sessão for SECUSER,
  --  então a conexão está utilizando um usuário de aplicação
  if v_user != 'secuser' then
   null;
  else
   v_user := lower(sys_context('userenv','proxy_user'));
  end if;
  -- Capturar a id do funcionário
  begin
   select employee_id into v_employee_id
   from hr.employees where lower(email)=lower(v_user);
  exception
   when no_data_found then
     v_employee_id:=0;
  end;
  -- Gerar Cláusula WHERE
  out_string:=out_string||' or account_mgr_id = '||v_employee_id;
 return out_string;
end;
```

Figura 3 – Exemplo de função para uma política VPD de controle de acesso

O segundo passo é conceder permissão de execução da função criada *f_policy_customers* para todos os usuários (comando *grant* da Figura 5) e, em seguida, aplicar a função como uma política da tabela, através da execução do método *dbms_rls.add_policy*. Na Figura 5 os parâmetros passados para o método *dbms_rls.add_policy* são, respectivamente:

- Nome do esquema da tabela onde será aplicada a política (*oe*);
- Nome da tabela onde será aplicada a política (*customers*);
- Nome da política (*accesscontrol_customers*), ou seja, nome atribuído à política criada, o qual é armazenado nos metadados do banco de dados e facilita a visualização da política na ferramenta de gestão (Figura 4), que pode ser acessada no caminho Iniciar > Programas > Oracle – [Nome da base de dados] > Ferramentas de Gerenciamento Integrado > Policy Manager;

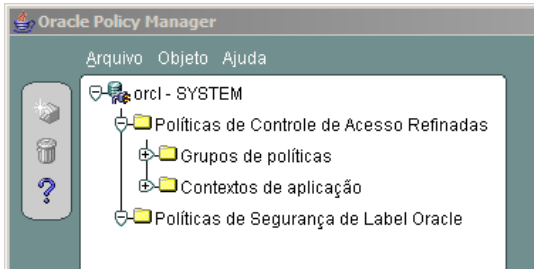


Figura 4 - Policy Manager, ferramenta de gestão de controle de acesso do Oracle

- Nome do esquema da função da política (*system*), ou seja, o nome do esquema ao qual a política pertence. É recomendado que seja criado um esquema para armazenar as funções e tabelas criados para tratar as regras de autorização, ou seja, separado do esquema onde estão os dados aos quais o acesso é controlado;
- Nome da função da política (*f_policy-customers*);
- Nome das colunas relevantes para a política (*CREDIT_LIMIT*). Se nenhuma coluna for especificada como relevante, a restrição de acesso será ativada para qualquer consulta na tabela.

```
grant execute on f_policy_customers to public;
begin
dbms_rls.add_policy('oe','customers','accesscontrol_customers',
 'system','f_policy_customers',sec_relevant_cols=>'CREDIT_LIMIT');
end;
```

Figura 5 – Exemplo de aplicação de uma função como uma política VPD

Os valores dos parâmetros do método *dbms_rls.add_policy* não precisam necessariamente serem passados na ordem definida. Para isso basta que a relação dos parâmetros com os seus valores seja feita explicitamente, como ilustra a Figura 6.

```
begin
dbms_rls.add_policy(
  object_schema => 'oe',
  object_name => 'customers',
  policy_name => 'accesscontrol_customers',
  function_schema => 'system',
  policy_function => 'f_policy_customers',
  sec_relevant_cols => 'CREDIT_LIMIT');
end;
```

Figura 6 – Exemplo de aplicação de uma função como uma política VPD explicitando os parâmetros do método *dbms_rls.add_policy*

Agora que a política foi definida e aplicada à tabela, se repetirmos a consulta apresentada na Figura 1, o resultado será diferente como mostra a Figura 7. Neste caso, o número de registros retornado pela consulta reduziu drasticamente, isso acontece porque agora, a funcionária *Eleni* pode enxergar apenas os seus próprios clientes. Quando a consulta foi aplicada sem restrição, foram retornadas 319 linhas (Figura 2), e quando a restrição foi aplicada, foram retornadas 74 linhas (Figura 7).

```
Krige                Clara                2300                 149
Chapman              Ian                  2400                 149
Garcia               Emmet                3600                 149
Barkin               Don                  5000                 149

74 rows selected.

SQL> set echo off
SQL>
```

Figura 7 - Exemplo de resultado da consulta com controle de acesso

Observe que, caso a consulta não inclua a coluna "credit_limit" (Figura 6), serão retornados todos os registros que satisfazem a consulta sem considerar a restrição de acesso, como pode ser visto na Figura 9. Isto ocorre porque somente a coluna "credit_limit" foi especificada como relevante.

```
connect "ELENI.ZLOTKEY@OSRD.COM"/welcome1
select cust_last_name, cust_first_name, account_mgr_id
 from oe.customers
 order by account_mgr_id;
```

Figura 8 - Exemplo de consulta sem a coluna relevante para o controle de acesso

```
Stockwell            Cary                      149
Olin                 Cary                      149
Ganesan              Clara                     149
Andrews              Ajay                      149
Prashant             Kathy                     149
Neeson               Graham                    149

319 rows selected.

SQL> set echo off
SQL>
```

Figura 9 - Exemplo de resultado da consulta sem a coluna relevante para o controle de acesso

2.1.2.3 Aplicando políticas VPD em colunas relevantes utilizando máscaras

Considere agora outro exemplo no qual os funcionários podem visualizar os dados dos clientes dos seus colegas, contudo, o limite de crédito do cliente continua exclusivo para o gerente da conta. Nesse caso, quando um funcionário tenta acessar o limite de crédito de um cliente de outro funcionário, o retorno da consulta será vazio.

A função utilizada nesse exemplo está representada na Figura 3, e é a mesma do exemplo anterior, a diferença está no momento de aplicar a função como política da tabela, assim como mostra a Figura 10. Neste caso, o procedimento "dbms_rls.add_policy" recebe um novo parâmetro: "sec_relevant_cols_opt => dbms_rls.ALL_ROWS", indicando que todas as linhas devem ser exibidas, e os dados protegidos serão mascarados, retornando NULL em seu lugar. É importante salientar que nesse caso, obrigatoriamente deverão ser definidas as colunas relevantes, pois serão essas colunas que terão seus dados mascarados. Além disso, a única forma de mascaramento possível é NULL. Não é permitido especificar outra *string* para substituição como, por exemplo, asterisco ("*****").

```
begin
dbms_rls.add_policy('oe','customers','accesscontrol_customers',
  'system','f_policy_customers',sec_relevant_cols=>'CREDIT_LIMIT',
  sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
end;
```

Figura 10 - Exemplo de aplicação de uma função como uma política VPD utilizando máscaras

Repetindo mais uma vez a consulta apresentada na Figura 1, obtemos o resultado apresentado na Figura 11. Observe que novamente foram selecionadas todas as linhas da tabela, mas se rolarmos a tela para o "account_mgr_id = 148" como mostra a Figura 12, vemos que o limite de crédito foi ocultado.

```
Olin                 Cary                 2300                 149
Ganesan              Clara                2300                 149
Andrews              Ajay                 2300                 149
Prashant             Kathy                2400                 149
Neeson               Graham               2400                 149

319 rows selected.

SQL> set echo off
SQL>
```

Figura 11 - Exemplo de resultado da consulta com controle de acesso utilizando mácaras

```
Sharif               Bob                                       148
Dench                Billy                                     148
Dickinson            Bo                                        148
Mastroianni          Blake                                     148
Seignier             Blake                1200                 149
Sen                  Meg                  3700                 149
Powell               Claude               1200                 149
Glenn                Faye                 1200                 149
```

Figura 12 - Exemplo de máscara no resultado da consulta com controle de acesso

2.1.2.4 Visualizando políticas

Quando uma política é aplicada a sua cláusula WHERE é armazenada na tabela v$vpd_policy. A Figura 13 apresenta uma consulta à tabela v$vpd_policy, que retorna as políticas que foram aplicadas.

```
Connected.
SQL> select policy, predicate as "predicate (WHERE clause)" from v$vpd_policy;

POLICY
------------------------------
predicate (WHERE clause)
--------------------------------------------------------------------------------
XDB$CONFIG$xd_sp


ACCESSCONTROL_CUSTOMERS
1=2  or account_mgr_id = 149


SQL>
```

Figura 13 – Exemplo de consulta à tabela v$vpd_policy

2.2 Label Security

2.2.1 Descrição da funcionalidade

O Oracle Label Security é utilizado para controlar o acesso às linhas das tabelas de um banco de dados, rotulando cada linha e cada usuário e concedendo ou negando o acesso de acordo com esses rótulos. Os rótulos, que podem ser personalizados, 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 etc. 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.

Como mencionado anteriormente, tanto as linhas quanto os usuários possuem rótulos. Para acessar uma linha protegida, as três regras a seguir devem ser verdadeiras:

1. O nível do rótulo do usuário deve ser igual ou superior ao nível do rótulo da linha.
2. Se forem definidos compartimentos para o rótulo da linha, então o usuário deve estar associado a todos os compartimento definidos para o rótulo da linha. Ou seja, se o usuário não estiver em um dos compartimentos, então o usuário não tem acesso.
3. Se forem definidos grupos para o rótulo da linha, então o usuário deve estar associado a pelo menos um dos grupos do rótulo da linha. Ou seja, se o usuário estiver em um grupo ele já tem acesso ao rótulo.

Os rótulos dos usuários podem ser gerenciados de maneira centralizada pelo Oracle Policy Manager ou localmente em cada banco de dados.

O Oracle Label Security fornece um sistema de políticas para definir os rótulos de dados e de usuários, que permite a criação de políticas específicas para o negócio e ainda a criação de múltiplas políticas em um mesmo banco de dados, dessa forma é possível criar políticas para várias aplicações em um só ambiente (Figura 14). Os rótulos podem ser inseridos como colunas ocultas nas tabelas, permitindo que declarações pré-existentes continuem funcionando.

Uma restrição importante em relação ao Label Security é que apenas uma política de segurança pode ser utilizada para proteger qualquer tabela. Ou seja, se forem atribuídas mais de uma política à mesma tabela, poderão ocorrer conflitos entre as políticas.

Oracle Label Security Data Label Components	Human Resources	Law Enforcement	Government
Levels	Confidential Sensitive Highly Sensitive	Level 1 Level 2 Level 3	Confidential Secret Top Secret
Compartments	PII Data Investigation	Internal Affairs Drug Enforcement	Desert Storm Border Protection
Groups	HR	DOJ FBI	NATO Homeland Security

Figura 14 – Exemplo de políticas do Oracle Label Security

2.2.2 Uso da funcionalidade

Essa seção apresenta exemplos práticos de controle de acesso com o Label Security, os quais se encontram nos tutoriais que compõem a OBE (Oracle By Example) [Oracle, 2009].

No cenário a seguir serão utilizadas as tabelas LOCATIONS e JOB_HISTORY e duas políticas serão criadas: FACILITY e PRIVACY. As políticas serão designadas às colunas FACLAB e PRIVLAB respectivamente, que por sua vez serão marcadas como ocultas.

2.2.2.1 Criando os usuários e políticas

O primeiro passo será criar os usuários e as políticas que serão utilizadas pelo Label Security, a Figura 15 apresenta os comandos utilizados para a criação dos usuários e para atribuição de privilégios. A Figura 16 apresenta os comandos para criação das políticas, com seus níveis, grupos e rótulos. Inicialmente são exibidos os comandos para criação da política FACILITY. Observe que na criação do grupo, foi criado uma hierarquia, na qual os grupos "United States", "Europe" e "Asia"herdam do grupo "Global".

```
CONNECT system/oracle
GRANT CREATE SESSION to MYCO_EMP IDENTIFIED BY MYCO_EMP;
GRANT CREATE SESSION to MYCO_MGR IDENTIFIED BY MYCO_MGR;
GRANT CREATE SESSION to MYCO_PLANNING IDENTIFIED BY MYCO_PLANNING;
CONNECT HR/hr
GRANT SELECT ON JOB_HISTORY TO MYCO_EMP;
GRANT SELECT ON JOB_HISTORY TO MYCO_MGR;
GRANT SELECT ON JOB_HISTORY TO MYCO_PLANNING;
GRANT SELECT ON LOCATIONS TO MYCO_EMP;
GRANT SELECT ON LOCATIONS TO MYCO_MGR;
GRANT SELECT, INSERT, UPDATE, DELETE ON LOCATIONS TO MYCO_PLANNING;
```

Figura 15 – Comandos para criação dos usuários do exemplo

```
CONNECT lbacsys/lbacsys
EXECUTE SA_SYSDBA.DROP_POLICY('FACILITY',TRUE);
EXECUTE SA_SYSDBA.DROP_POLICY('PRIVACY',TRUE);
-- Criar a política FACILITY
EXECUTE SA_SYSDBA.CREATE_POLICY('FACILITY','FACLAB',
 'READ_CONTROL,CHECK_CONTROL,LABEL_DEFAULT,HIDE');
-- Adicionar níveis de sensibilidade à política FACILITY
EXECUTE SA_COMPONENTS.CREATE_LEVEL('FACILITY',1000,'P','PUBLIC');
EXECUTE SA_COMPONENTS.CREATE_LEVEL('FACILITY',2000,'S','SENSITIVE');
EXECUTE SA_COMPONENTS.CREATE_LEVEL('FACILITY',3000,'HS','HIGHLY_SENSITIVE');
-- Adicionar grupos à política FACILITY
EXECUTE SA_COMPONENTS.CREATE_GROUP('FACILITY',1000,'Global','Global');
EXECUTE SA_COMPONENTS.CREATE_GROUP('FACILITY',101,'US','United States','GLOBAL');
EXECUTE SA_COMPONENTS.CREATE_GROUP('FACILITY',102,'EU','Europe','GLOBAL');
EXECUTE SA_COMPONENTS.CREATE_GROUP('FACILITY',103,'Asia','Asia','GLOBAL');
-- Criar rótulos para a política FACILITY
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',1000,'P');
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',2101,'S::US');
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',3101,'HS::US');
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',2103,'S::ASIA');
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',3103,'HS::ASIA');
-- Criar a política PRIVACY
EXECUTE SA_SYSDBA.CREATE_POLICY('PRIVACY','PRIVLAB',
 'READ_CONTROL,CHECK_CONTROL,LABEL_DEFAULT,HIDE');
-- Adicionar níveis de sensibilidade à política PRIVACY
EXECUTE SA_COMPONENTS.CREATE_LEVEL('PRIVACY',1000,'C','CONFIDENTIAL');
EXECUTE SA_COMPONENTS.CREATE_LEVEL('PRIVACY',2000,'S','SENSITIVE');
-- Criar rótulos para a política PRIVACY
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL ('PRIVACY',101000,'C');
EXECUTE SA_LABEL_ADMIN.CREATE_LABEL ('PRIVACY',102000,'S');
```

Figura 16 – Comandos para criação das políticas

Agora que as políticas foram criadas, elas podem ser revisadas e alteradas através do Oracle Policy Manager, que pode ser acessado através do Oracle Enterprise Manager. O Oracle Policy Manager apresenta uma árvore à esquerda, que exibe as políticas do Label security, e uma série de abas à direita. Primeiramente, vamos checar a política FACILITY, a Figura 17 apresenta os níveis de sensibilidade da política, a Figura 18 apresenta os grupos criados para essa política e, finalmente, a Figura 19 apresenta os rótulos que foram criados. Agora para a política PRIVACY, a Figura 20 apresenta níveis de sensibilidade criados enquanto a Figura 21 apresenta os rótulos criados.

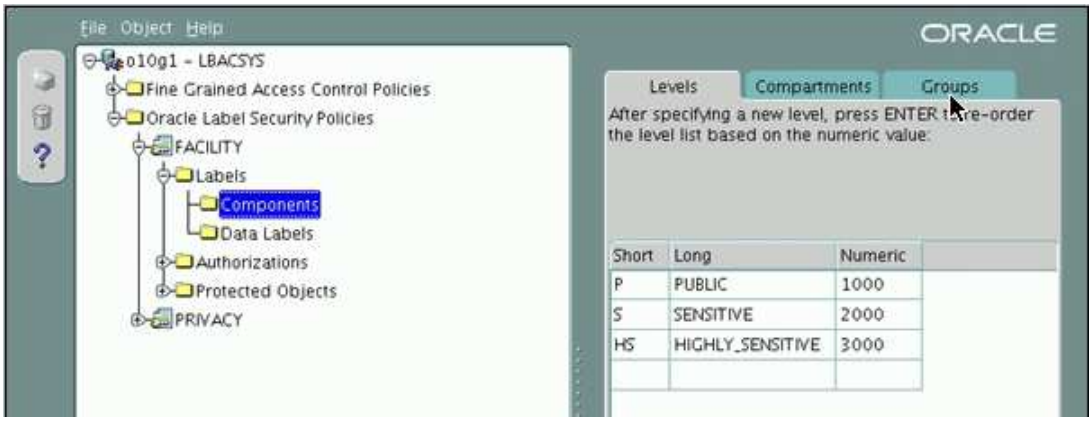


Figura 17 – Níveis de sensibilidade da política FACILITY visualizados pelo Oracle Policy Manager

Figura 18 - Grupos da política FACILITY visualizados pelo Oracle Policy Manager

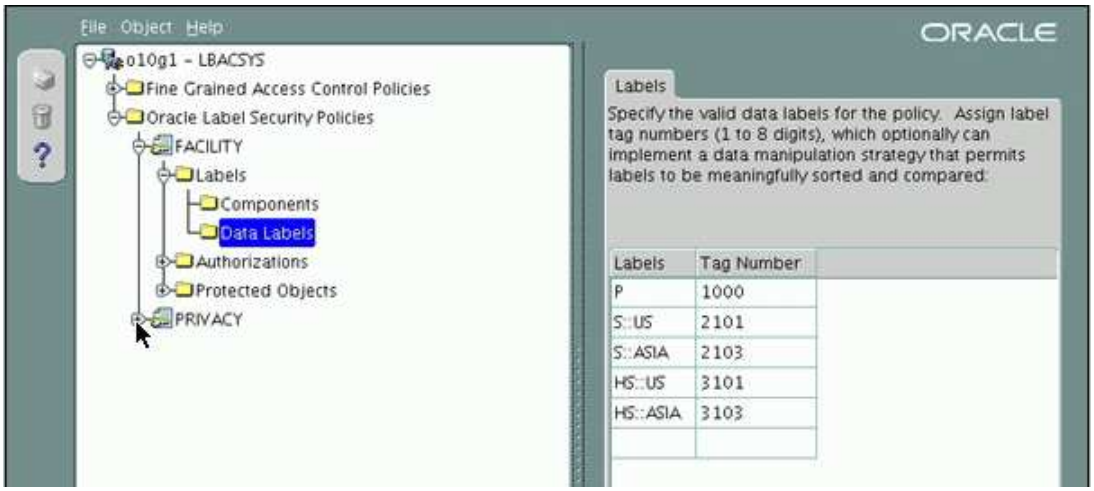


Figura 19 - Rótulos da política FACILITY visualizados pelo Oracle Policy Manager

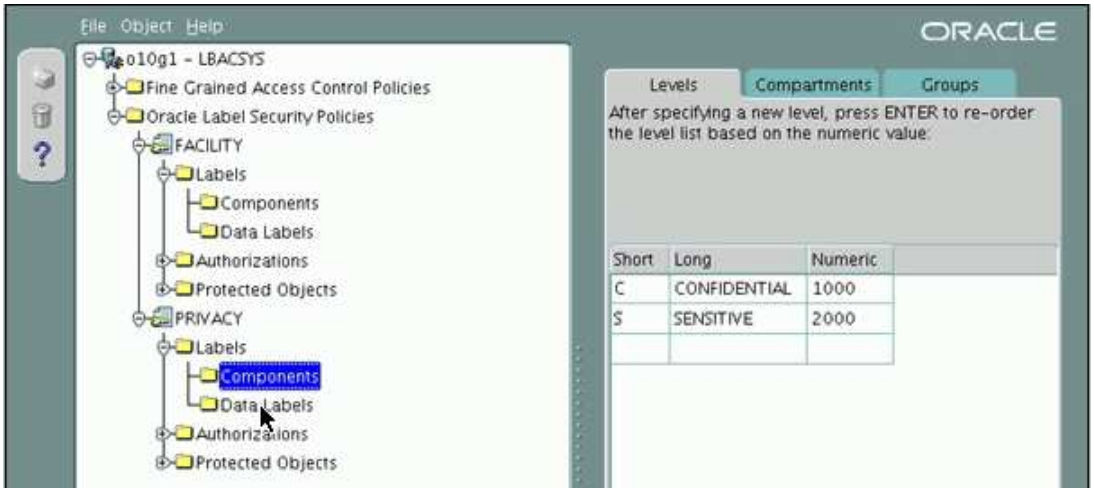


Figura 20 - Níveis de sensibilidade da política PRIVACY visualizados pelo Oracle Policy Manager

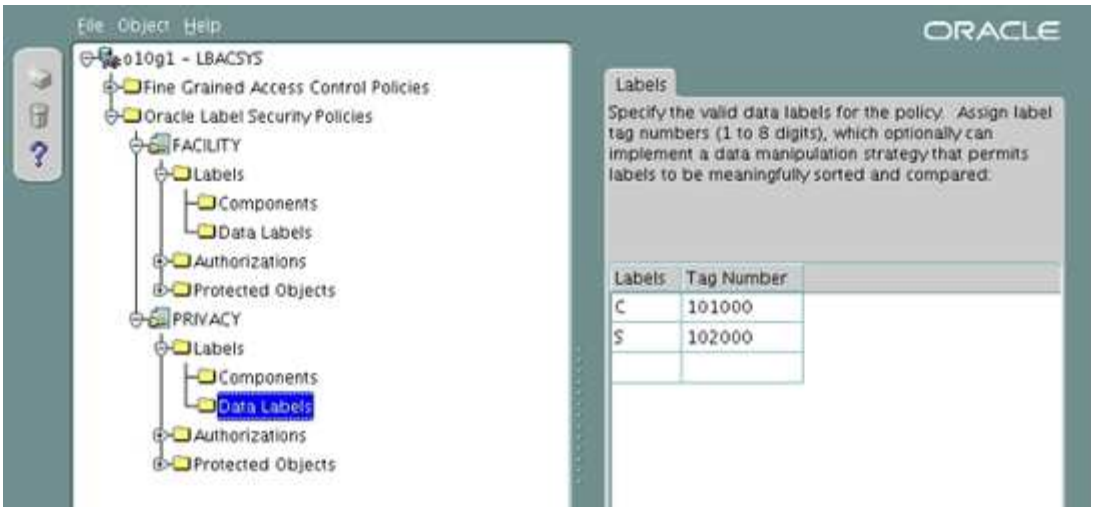


Figura 21 – Rótulos da política PRIVACY visualizados pelo Oracle Policy Manager

Existe um usuário padrão para criação e manutenção das políticas, o LBACSYS. Após a criação de uma política, o perfil de administração da política é automaticamente criado ("<policy_name>_DBA") e dado ao LBACSYS que por sua vez pode autorizar outros usuários para manutenção da política através de comandos "grant".

2.2.2.2 Aplicando autorização aos usuários

Agora que todas as políticas já foram criadas e testadas o próximo passo é aplicar as autorizações aos usuários (Figura 22). O usuário HR precisa de acesso total de leitura e escrita para todos os dados, além de poder alterar rótulos de sessão e privilégios de sessão de outros usuários. Isto foi feito nos dois últimos comandos da Figura 22.

A Figura 22 apresenta os comandos necessários para definição das autorizações para os respectivos usuários:

- Ao usuário MYCO_EMP foi atribuído nível de autorização PUBLIC para a política Facility, a qual pode ser visualizada no Oracle Policy Manager, como apresentado na Figura 23. Como foi atribuído apenas um nível de autorização para este usuário, este nível é o nível mínimo, máximo, default e por linha na inserção. O atributo *Row* (*Row level on INSERT*) é o nível que será atribuído aos registros inseridos pelo usuário. Este nível deve ser maior ou igual ao mínimo e menor ou igual ao nível definido como default. Por exemplo, se o mínimo do usuário é 3 e o máximo do usuário é 7 e o default é 4, então o usuário pode atribuir os níveis 3 e 4. Ou seja, ele não poderia atribuir um nível 6, por exemplo. O usuário só pode ter acesso aos registros cujos rótulos estão entre o nível mínimo e o nível máximo a ele atribuído;
- Ao usuário MYCO_MGR foi atribuída nível de autorização CONFIDENTIAL em relação à política PRIVACY (Figura 24) e acesso SENTIVE e PUBLIC na política FACILITY (Figura 25). Observe que o nível de acesso de MYCO_MGR para a política FACILITY, varia de no mínimo PUBLIC para no máximo SENSITIVE.
  Além disso, vale ressaltar que a sintaxe do SET_USER_LABELS é "**nível de acesso:compartimentos autorizados:grupos autorizados".** Logo, a string "S::US,EU,ASIA" significa que o usuário tem nível de acesso SENSITIVE nos grupos US, EU e ASIA, como nenhum compartimento foi criado não houve necessidade de especificar o valor para este parâmetro.

- Ao usuário MYCO_PLANNING foi atribuído nível de autorização HIGHLY_SENSITIVE do grupo GLOBAL (Figura 26);

Os níveis máximo de escrita, mínimo de escrita, default e de linha também podem ser definidos. Nestes exemplos, eles não foram especificados. Dessa forma, eles assumiram valores baseados no nível máximo de leitura do usuário e no nível mínimo da política.

```
CONNECT lbacsys/lbacsys
-- Configurar autorizações dos usuários MYCO_EMP, MYCO_MGR e MYCO_PLANNING
EXECUTE SA_USER_ADMIN.SET_USER_LABELS ('PRIVACY','MYCO_MGR','C');
EXECUTE SA_USER_ADMIN.SET_USER_LABELS ('FACILITY','MYCO_EMP','P');
EXECUTE SA_USER_ADMIN.SET_USER_LABELS ('FACILITY','MYCO_MGR','S::US,EU,ASIA');
EXECUTE SA_USER_ADMIN.SET_USER_LABELS ('FACILITY','MYCO_PLANNING','HS::GLOBAL');
-- Privilégios do usuário HR
-- FULL - Leitura e escrita para todos os dados
-- PROFILE_ACCESS - Alterar rótulos e privilégios de sessão
EXECUTE SA_USER_ADMIN.SET_USER_PRIVS ('PRIVACY','HR','FULL,PROFILE_ACCESS');
EXECUTE SA_USER_ADMIN.SET_USER_PRIVS ('FACILITY','HR','FULL,PROFILE_ACCESS');
```

Figura 22 – Comandos para aplicação das autorizações

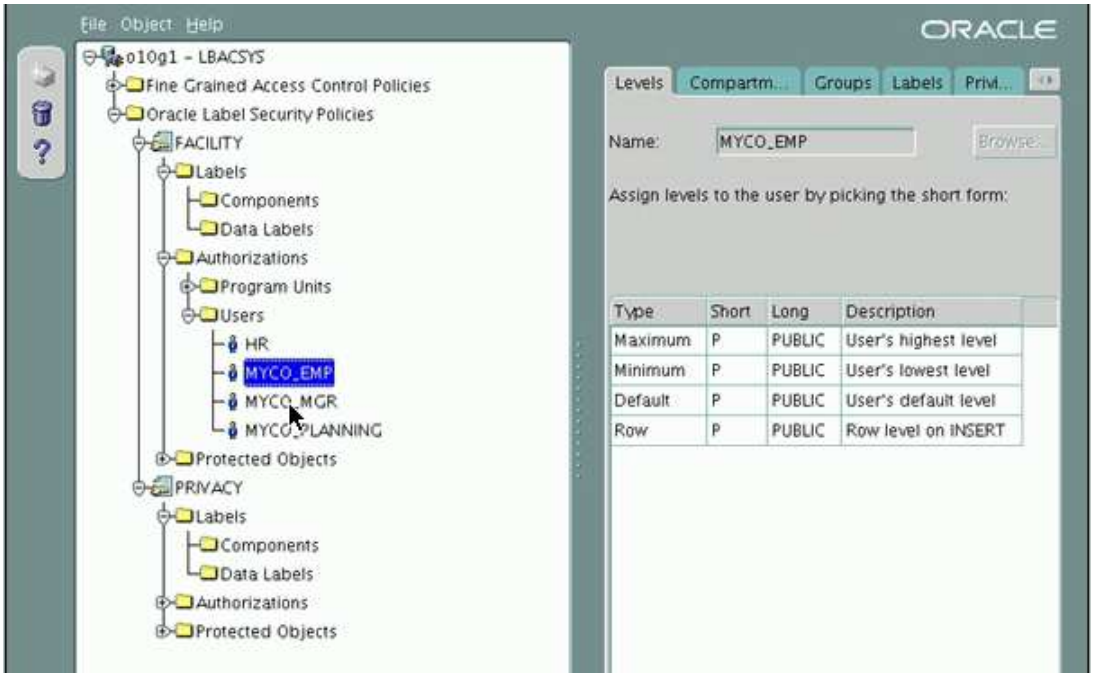


Figura 23 - Níveis de acesso da política FACILITY para o usuário MYCO_EMP visualizados pelo Oracle Policy Manager

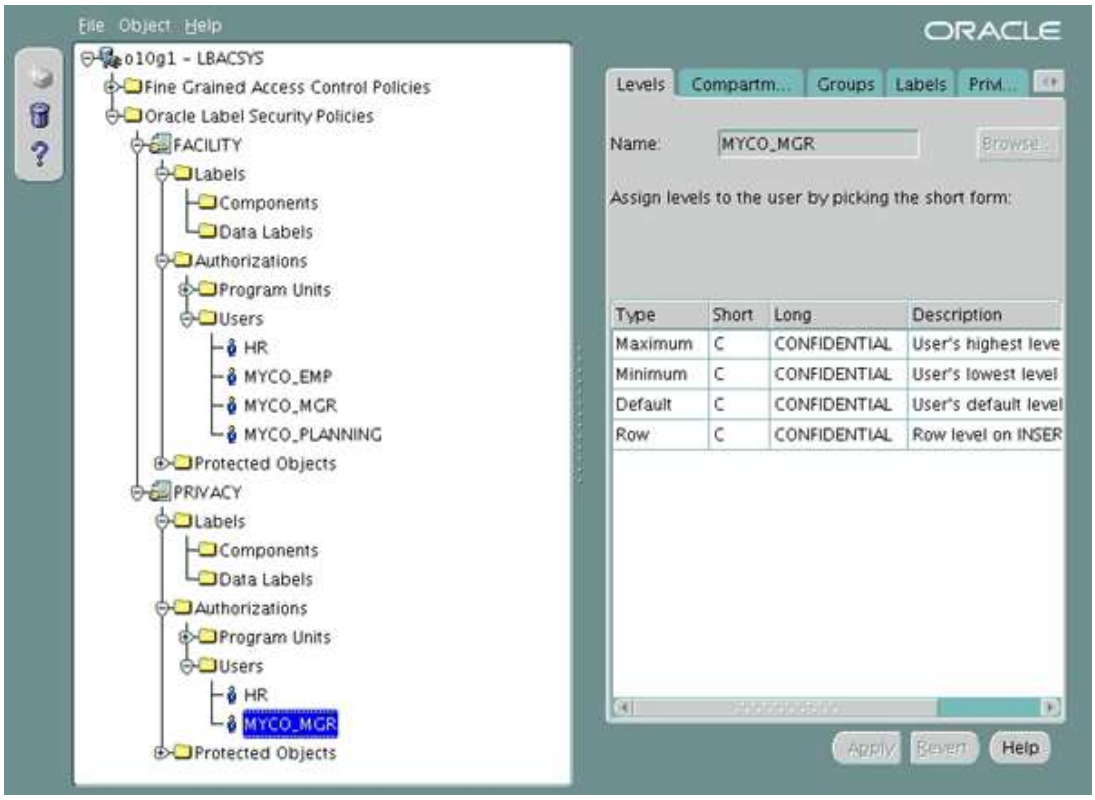


Figura 24 - Níveis de acesso da política PRIVACY para o usuário MYCO_MGR visualizados pelo Oracle Policy Manager

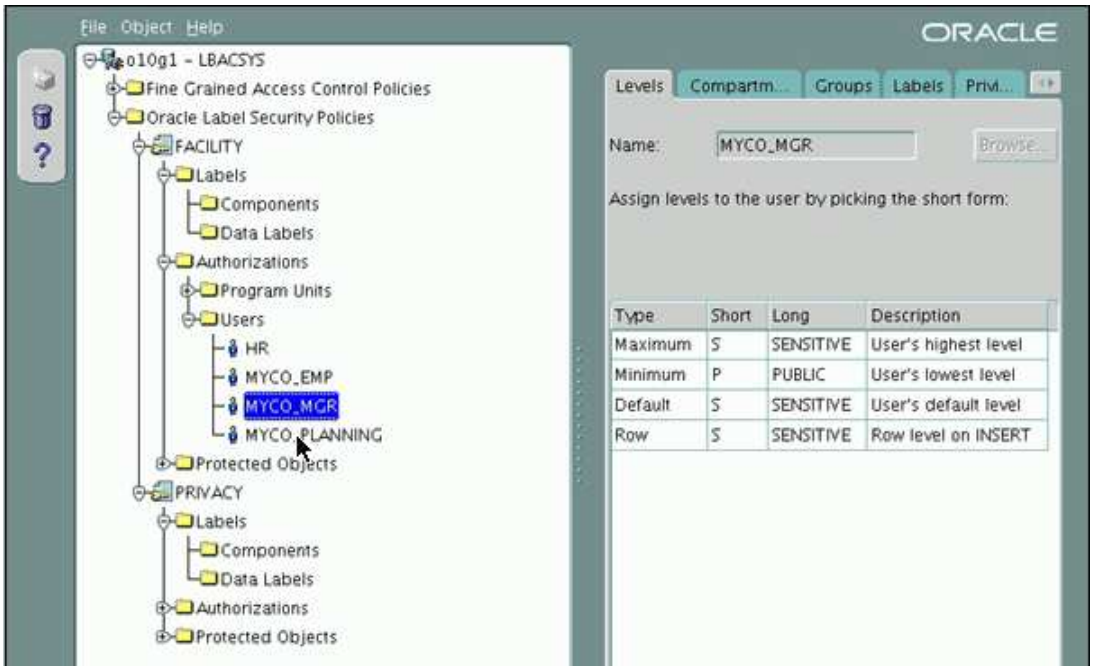


Figura 25 - Níveis de acesso da política FACILITY para o usuário MYCO_MGR visualizados pelo Oracle Policy Manager

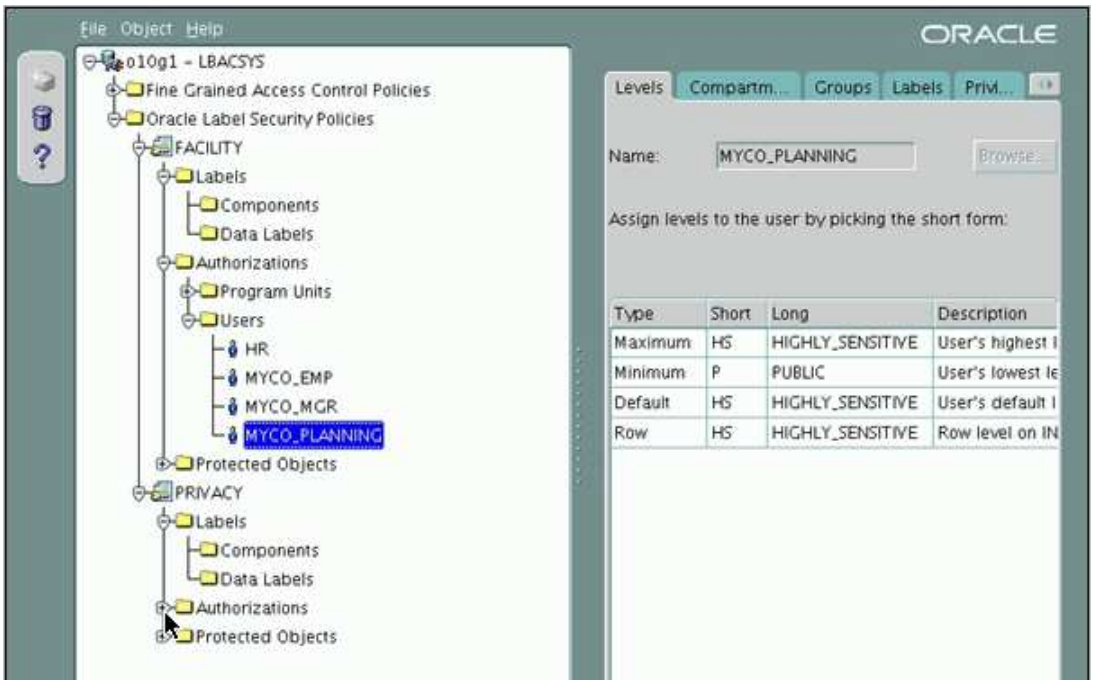


Figura 26 - Níveis de acesso da política FACILITY para o usuário MYCO_PLANNING visualizados pelo Oracle Policy Manager

2.2.2.3 Aplicado as políticas às tabelas

As políticas do Label Security podem ser aplicadas em um schema inteiro ou em tabelas individuais, nós aplicaremos nas tabelas LOCATIONS e JOB_HISTORY através dos comandos representados na Figura 27 e na Figura 28 respectivamente.

```
BEGIN
   sa_policy_admin.apply_table_policy (
     POLICY_NAME => 'FACILITY',
     SCHEMA_NAME => 'HR',
     TABLE_NAME  => 'LOCATIONS',
     TABLE_OPTIONS => NULL,
     LABEL_FUNCTION => NULL,
END;
```

Figura 27 - Comandos para aplicação da política FACILITY na tabela LOCATIONS

```
BEGIN
   sa_policy_admin.apply_table_policy (
     POLICY_NAME => 'PRIVACY',
     SCHEMA_NAME => 'HR',
     TABLE_NAME  => 'JOB_HISTORY',
     TABLE_OPTIONS => NULL,
     LABEL_FUNCTION => NULL,
END;
```

Figura 28 - Comandos para aplicação da política PRIVACY na tabela JOB_HISTORY

Com as políticas aplicadas às tabelas, podemos revisá-las através do Oracle Policy Manager. A Figura 29 apresenta a política FACILITY aplicada à tabela LOCATIONS, enquanto a Figura 30 apresenta a política PRIVACY aplicada à tabela JOB_HISTORY. A aba *Options* apresenta os controles que podem ser definidos para a aplicação da política na tabela. Dessa forma, pode-se definir que a política deve ser aplicada para leitura, escrita (insert, update e delete) e o controle de escrita de label quando dados são inseridos e atualizados.

- "LABEL_DEFAULT" marcado define que quando o usuário não especificar um rótulo para ser atribuído ao registro, o rótulo default do usuário será utilizado. Se "LABEL_DEFAULT" não estiver marcado e nem uma função para rotulação da linha estiver definida e o usuário tentar inserir uma linha, então um erro ocorre.
- "LABEL_UPDATE" marcado define que apenas usuários com privilégios de WRITEUP, WRITEDOWN e WRITEACROSS poderão atualizar o rótulo de uma linha. Usualmente, quando "LABEL_UPDATE" não está marcado, o usuário pode atualizar o rótulo de uma linha para qualquer valor dentro do intervalo de *label* ao qual ele está autorizado. Estes privilégios de usuário só são ativados quando o LABEL_UPDATE está ativado.
    - O privilégio WRITEUP autoriza usuários a aumentar o nível de um rótulo de um registro até o nível máximo definido para o usuário.
    - O privilégio WRITEDOWN autoriza usuários a diminuir o nível de um rótulo de um registro para qualquer nível maior ou igual ao nível mínimo definido para o usuário.
    - O privilégio WRITEACROSS autoriza usuários a mudar grupos e compartimentos de um rótulo de um registro, mas não autoriza mudança de nível.
- "CHECK_CONTROL" garante que o usuário conseguirá ler um registro após a inserção do registro com o rótulo atribuído ao mesmo. Este controle deve ser usado quando o usuário não pode inserir um registro atribuindo um rótulo que está fora do intervalo de rótulos que ele possui. Logo, com "CHECK_CONTROL", o usuário só poderá inserir um registro com um rótulo dentro deste intervalo, senão será levantado um erro. Sem este campo marcado, o usuário poderá atribuir um rótulo ao registro que faz com que este não possa ser mais acessado pelo usuário depois da execução da operação.

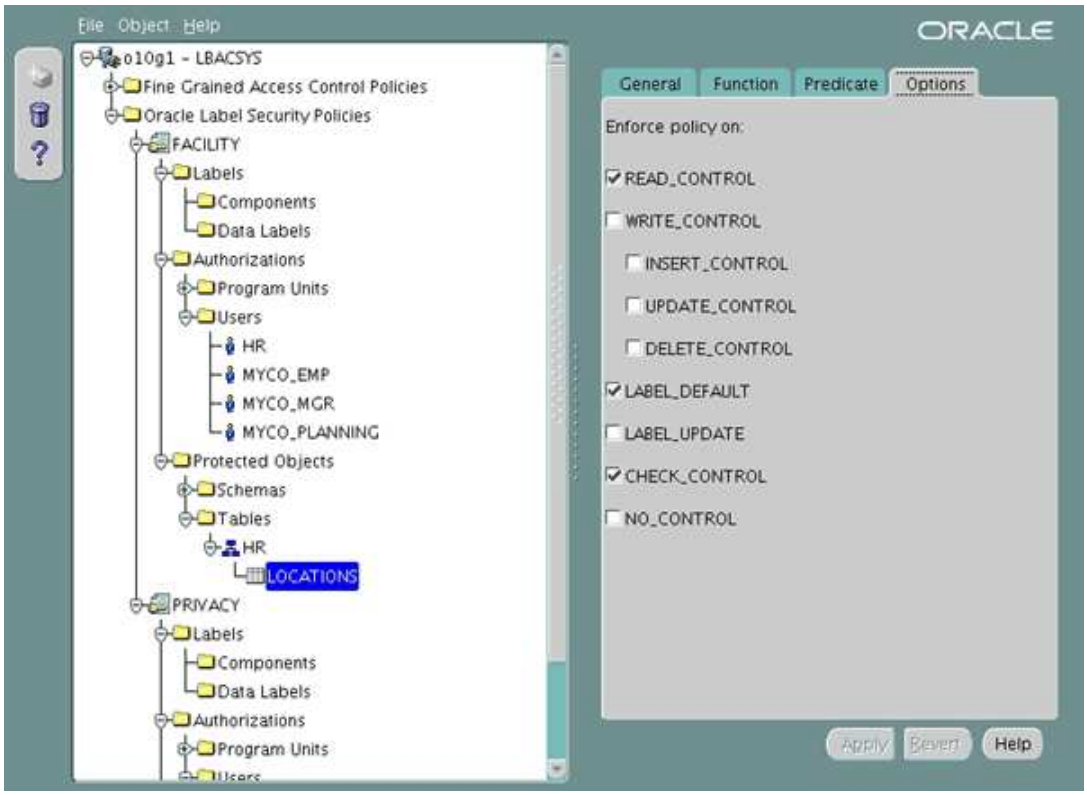


Figura 29 – Política FACILITY aplicada à tabela LOCATIONS visualizada pelo Oracle Policy Manager

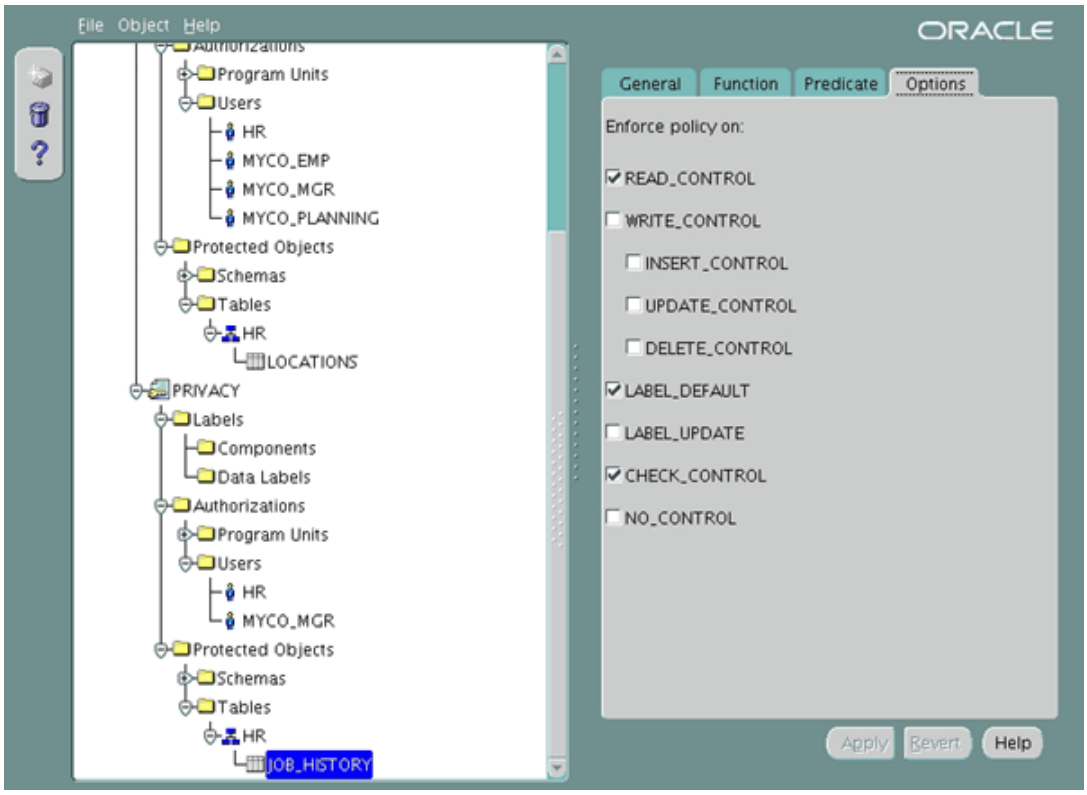


Figura 30 - Política PRIVACY aplicada à tabela JOB_HISTORY visualizada pelo Oracle Policy Manager

2.2.2.4 Adicionando rótulos aos dados

O útimo passo a ser realizado antes de testarmos o controle de acesso é adicionar os rótulos aos dados das tabelas. A Figura 31 apresenta os comandos necessários para aplicar os rótulos aos dados.

```
-- Adicionar rótulos da política FACILITY à locais específicos na Asia
UPDATE hr.locations set faclab = char_to_label('FACILITY','S::ASIA')
WHERE upper(city) in ('BEIJING','TOKYO','SINGAPORE');
-- Adicionar rótulos da política FACILITY à locais específicos nos EUA
UPDATE hr.locations set faclab = char_to_label('FACILITY','HS::US')
WHERE upper(city) in ('SOUTH SAN FRANCISCO');
-- Adicionar rótulos da política FACILITY aos locais restantes
UPDATE hr.locations set faclab = char_to_label('FACILITY','P')
WHERE faclab is NULL;
--Adicionar rótulos da política PRIVACY
UPDATE hr.job_history set privlab = char_to_label('PRIVACY','S')
WHERE ((to_char(sysdate,'YYYY') - to_char(end_date,'YYYY')) > 7);
UPDATE hr.job_history set privlab = char_to_label('PRIVACY','C')
WHERE ((to_char(sysdate,'YYYY') - to_char(end_date,'YYYY')) <= 7);
COMMIT;
```

Figura 31 – Comandos para adicionar rótulos aos dados

2.2.2.5 Testando o controle de acesso

Agora que todas as políticas foram definidas, todos os usuários receberam as suas autorizações e todos os dados foram rotulados, os testes de controle de acesso podem ser realizados. Primeiramente vamos testar a política FACILITY, a Figura 32 apresenta a consulta de teste de acesso para o usuário MYCO_EMP, e a Figura 33 apresenta o resultado desse teste.

```
CONNECT myco_emp/myco_emp
SELECT locations.*, label_to_char(faclab)
  "FACILITY LABEL" FROM hr.locations;
```

Figura 32 – Consulta de teste de acesso do usuário MYCO_EMP na tabela LOCATIONS

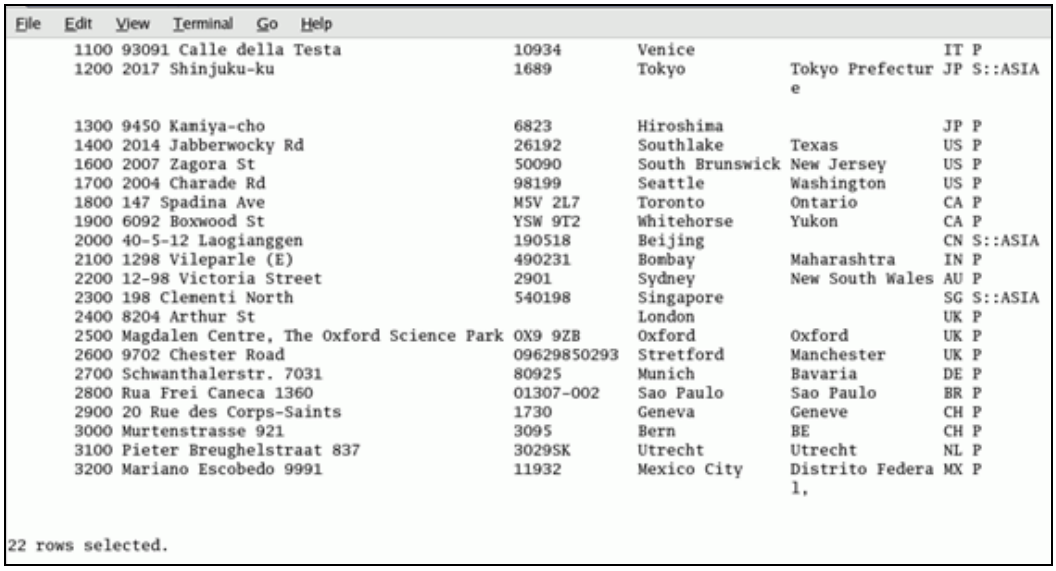
```
File  Edit  View  Terminal  Go  Help

LOCATION_ID STREET_ADDRESS                          POSTAL_CODE  CITY           STATE_PROVINCE  CO FACILITY LABEL
----------- --------------------------------------- -----------  -------------- --------------  -- ---------------
-----
       1000 1297 Via Cola di Rie                    00989        Roma                           IT P
       1100 93091 Calle della Testa                 10934        Venice                         IT P
       1300 9450 Kamiya-cho                         6823         Hiroshima                      JP P
       1400 2014 Jabberwocky Rd                     26192        Southlake      Texas           US P
       1600 2007 Zagora St                          50090        South Brunswick New Jersey     US P
       1700 2004 Charade Rd                         98199        Seattle        Washington      US P
       1800 147 Spadina Ave                         M5V 2L7      Toronto        Ontario         CA P
       1900 6092 Boxwood St                         YSW 9T2      Whitehorse     Yukon           CA P
       2100 1298 Vileparle (E)                      490231       Bombay         Maharashtra     IN P
       2200 12-98 Victoria Street                   2901         Sydney         New South Wales AU P
       2400 8204 Arthur St                                       London                         UK P
       2500 Magdalen Centre, The Oxford Science Park OX9 9ZB      Oxford         Oxford          UK P
       2600 9702 Chester Road                       09629850293  Stretford      Manchester      UK P
       2700 Schwanthalerstr. 7031                   80925        Munich         Bavaria         DE P
       2800 Rua Frei Caneca 1360                    01307-002    Sao Paulo      Sao Paulo       BR P
       2900 20 Rue des Corps-Saints                 1730         Geneva         Geneve          CH P
       3000 Murtenstrasse 921                       3095         Bern           BE              CH P
       3100 Pieter Breughelstraat 837               3029SK       Utrecht        Utrecht         NL P
       3200 Mariano Escobedo 9991                   11932        Mexico City    Distrito Federa MX P
                                                                                 l,


19 rows selected.

Hit Return To Continue
```

Figura 33 – Resultado da consulta de teste de acesso do usuário MYCO_EMP na tabela LOCATIONS

Nesse primeiro teste foram retornadas 19 linhas. Como o usuário MYCO_EMP recebeu apenas o acesso PUBLIC, ele visualiza um número restrito de entradas. Para o segundo teste a Figura 34 apresenta a consulta de teste de acesso para o usuário MYCO_MGR, e a Figura 35 apresenta o resultado desse teste. Observe que dessa vez 22 linhas forma retornadas. Como o usuário MYCO_MGR possui o acesso SENSITIVE, além das entradas PUBLIC ele também visualiza as entradas de Singapore, Beijing e Tokyo.

```
CONNECT myco_mgr/myco_mgr
SELECT locations.*, label_to_char(faclab)
  "FACILITY LABEL" FROM hr.locations;
```

Figura 34 – Consulta de teste de acesso do usuário MYCO_MGR na tabela LOCATIONS



File Edit View Terminal Go Help

```
      1100 93091 Calle della Testa                  10934       Venice                         IT P
      1200 2017 Shinjuku-ku                         1689        Tokyo          Tokyo Prefectur JP S::ASIA
                                                                               e

      1300 9450 Kamiya-cho                          6823        Hiroshima                      JP P
      1400 2014 Jabberwocky Rd                      26192       Southlake      Texas           US P
      1600 2007 Zagora St                           50090       South Brunswick New Jersey     US P
      1700 2004 Charade Rd                          98199       Seattle        Washington      US P
      1800 147 Spadina Ave                          M5V 2L7     Toronto        Ontario         CA P
      1900 6092 Boxwood St                          YSW 9T2     Whitehorse     Yukon           CA P
      2000 40-5-12 Laogianggen                      190518      Beijing                        CN S::ASIA
      2100 1298 Vileparle (E)                       490231      Bombay         Maharashtra     IN P
      2200 12-98 Victoria Street                    2901        Sydney         New South Wales AU P
      2300 198 Clementi North                       540198      Singapore                      SG S::ASIA
      2400 8204 Arthur St                                       London                         UK P
      2500 Magdalen Centre, The Oxford Science Park OX9 9ZB     Oxford         Oxford          UK P
      2600 9702 Chester Road                        09629850293 Stretford      Manchester      UK P
      2700 Schwanthalerstr. 7031                    80925       Munich         Bavaria         DE P
      2800 Rua Frei Caneca 1360                     01307-002   Sao Paulo      Sao Paulo       BR P
      2900 20 Rue des Corps-Saints                  1730        Geneva         Geneve          CH P
      3000 Murtenstrasse 921                        3095        Bern           BE              CH P
      3100 Pieter Breughelstraat 837                3029SK      Utrecht        Utrecht         NL P
      3200 Mariano Escobedo 9991                    11932       Mexico City    Distrito Federa MX P
                                                                               l,

22 rows selected.
```

Figura 35 – Resultado da consulta de teste de acesso do usuário MYCO_MGR na tabela LOCATIONS

Para o terceiro teste a Figura 36 apresenta a consulta de teste de acesso para o usuário MYCO_PLANNING, e a Figura 37 apresenta o resultado desse teste. Agora nesse terceiro teste foram retornadas 23 linhas, como o usuário MYCO_PLANNING possui o acesso HIGHLY-SENSITIVE ele pode visualizar ainda mais uma entrada, a de South San Francisco.

```
CONNECT myco_planning/myco_planning
SELECT locations.*, label_to_char(faclab)
  "FACILITY LABEL" FROM hr.locations;
```

Figura 36 – Consulta de teste de acesso do usuário MYCO_PLANNING na tabela LOCATIONS

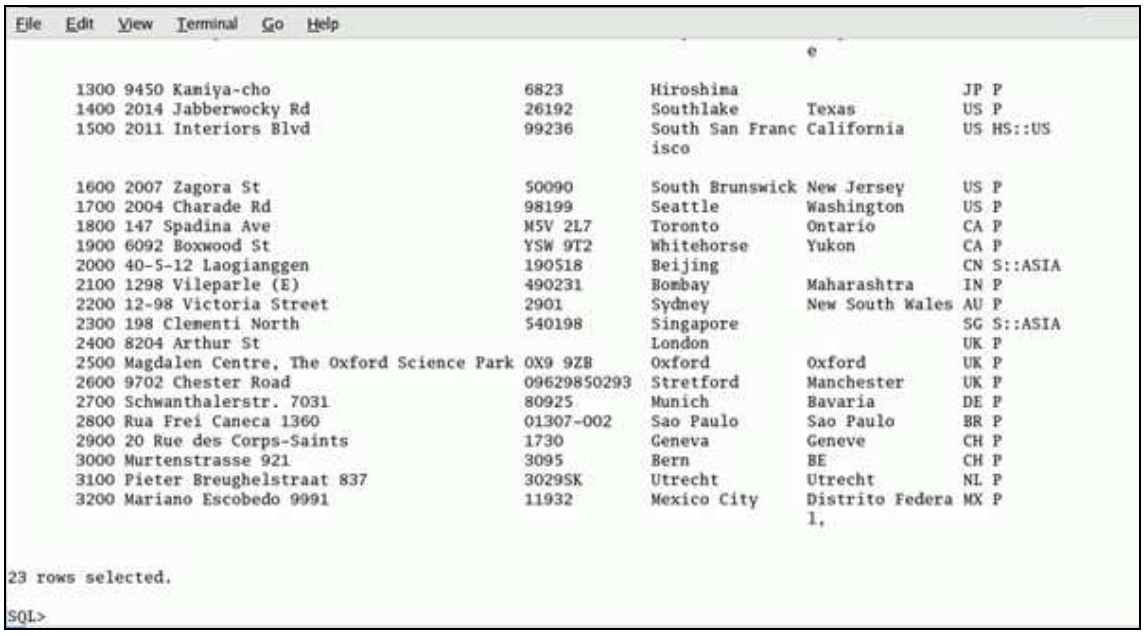


Figura 37 – Resultado da consulta de teste de acesso do usuário MYCO_PLANNING na tabela LOCATIONS

A próxima série de testes será realizada para a política PRIVACY, a Figura 38 apresenta a consulta de teste de acesso para o usuário MYCO_EMP, e a Figura 39 apresenta o resultado desse teste. Note que o usuário MYCO_EMP não pode visualizar nenhuma entrada pois não recebeu autorização nessa política.

```
CONNECT myco_emp/myco_emp
SELECT job_history.*, label_to_char(PRIVLAB)
  "PRIVACY LABEL" FROM hr.job_history;
```

Figura 38 – Consulta de teste de acesso do usuário MYCO_EMP na tabela JOB_HISTORY

```
SQL> @ols_test_privacy
******************************************************
* Connect to the Oracle database as Application User
* myco_emp
*
* select job_history.*, label_to_char(PRIVLAB)
*  "PRIVACY LABEL" from hr.job_history

******************************************************
Hit Return To Continue

Connected.

no rows selected

******************************************************
* Connect to the Oracle database as Application User
* myco_mgr
*
* select job_history.*, label_to_char(PRIVLAB)
*  "PRIVACY LABEL" from hr.job_history

******************************************************
Hit Return To Continue
```

Figura 39 – Resultado da consulta de teste de acesso do usuário MYCO_EMP na tabela JOB_HISTORY

Para o próximo teste a Figura 40 apresenta a consulta de teste de acesso para o usuário MYCO_MGR, e a Figura 41 apresenta o resultado desse teste. Observe que dessa vez o usuário MYCO_MGR pode ver somente as entradas onde o campo END_DATE tem menos de sete anos, pois recebeu exclusivamente o acesso CONFIDENTIAL.

```
CONNECT myco_mgr/myco_mgr
SELECT job_history.*, label_to_char(PRIVLAB)
  "PRIVACY LABEL" FROM hr.job_history;
```

Figura 40 – Consulta de teste de acesso do usuário MYCO_MGR na tabela JOB_HISTORY

```
File  Edit  View  Terminal  Go  Help
*  "PRIVACY LABEL" from hr.job_history

**************************************************
Hit Return To Continue

Connected.

no rows selected

**************************************************
* Connect to the Oracle database as Application User
* myco_mgr
*
* select job_history.*, label_to_char(PRIVLAB)
*  "PRIVACY LABEL" from hr.job_history

**************************************************
Hit Return To Continue

Connected.

EMPLOYEE_ID START_DAT END_DATE  JOB_ID     DEPARTMENT_ID PRIVACY LABEL
----------- --------- --------- ---------- ------------- ----------------
        201 17-FEB-96 19-DEC-99 MK_REP                20 C
        114 24-MAR-98 31-DEC-99 ST_CLERK              50 C
        122 01-JAN-99 31-DEC-99 ST_CLERK              50 C
        176 01-JAN-99 31-DEC-99 SA_MAN                80 C

Hit Return To Continue
```

Figura 41 – Resultado da consulta de teste de acesso do usuário MYCO_MGR na tabela JOB_HISTORY

Para o último teste, a Figura 42 apresenta a consulta de teste de acesso para o usuário HR, e a Figura 43 apresenta o resultado desse teste. Nesse último caso o usuário HR pode visualizar todos os dados da tabela.

```
CONNECT hr/hr@orcl
SELECT job_history.*, label_to_char(PRIVLAB)
  "PRIVACY LABEL" FROM hr.job_history;
```

Figura 42 – Consulta de teste de acesso do usuário HR na tabela JOB_HISTORY

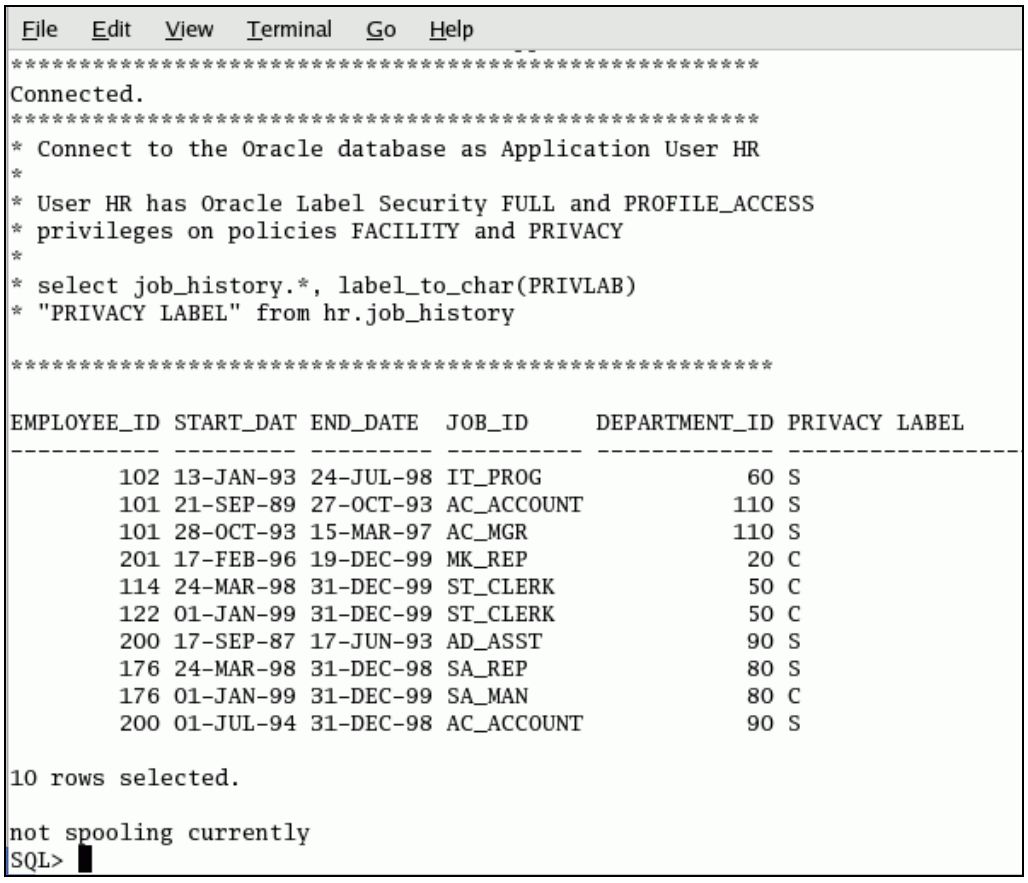


Figura 43 – Resultado da consulta de teste de acesso do usuário MYCO_MGR na tabela JOB_HISTORY

3 Benchmark utilizado para a realização dos testes das funcionalidades

O TPC-H é um benchmark de apoio à decisão [TPC-H, 2008, 2009]. Ele consiste de um conjunto de consultas *ad-hoc* orientadas ao negócio. As consultas e a população dos dados no banco de dados foram escolhidas para permitir uma grande relevância para toda a indústria. Esse benchmark ilustra sistemas de apoio à decisão que examinam um grande volume de dados, executam consultas com um alto grau de complexidade, e dão respostas para questões críticas do negócio. Dessa forma, o TPC-H consiste em um conjunto de consultas e dados que permite demonstrar a viabilidade da proposta.

3.1 Modelo de dados do TPC-H

O modelo de dados do TPC-H, considerando apenas os nomes das relações, é apresentado na Figura 44. Este modelo foi gerado a partir da ferramenta VisualTPCH[1] [Domingues *et al.*, 2008]. As arestas apontam na direção de relacionamentos um para muitos entre tabelas. Logo, uma *nation* possui vários *customers*. A Figura 45 apresenta o modelo do TPC-H incluindo os atributos das entidades. Os parênteses seguindo o nome da tabela contêm o prefixo dos nomes das colunas de cada tabela. O

[1] http://gbd.dc.ufscar.br/index.php?option=com_content&view=article&id=12&Itemid=21&lang=en

número/fórmula abaixo de cada nome de tabela representa a cardinalidade (número de linhas) da tabela. Alguns valores consideram o fator de escala SF (*Scale Factor*) para obter o tamanho da tabela escolhido. A cardinalidade para a tabela LINEITEM é aproximada. Maiores detalhes sobre o modelo de dados podem ser encontrados em http://www.tpc.org/tpch/.

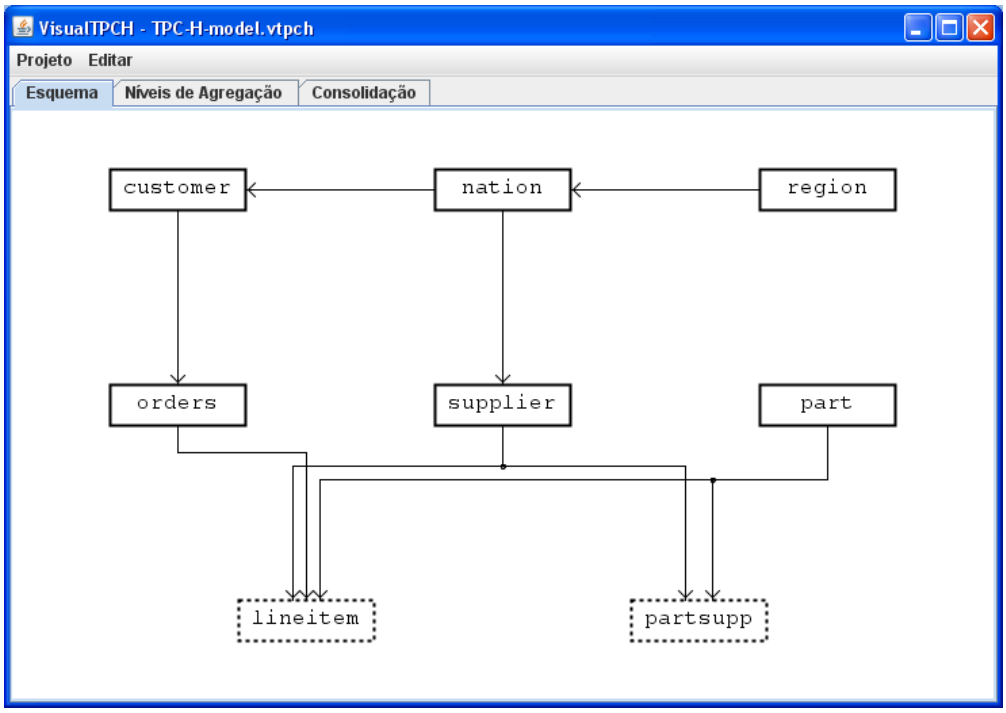


Figura 44 – Modelo de dados do TPC-H [Domingues *et al.*, 2008]

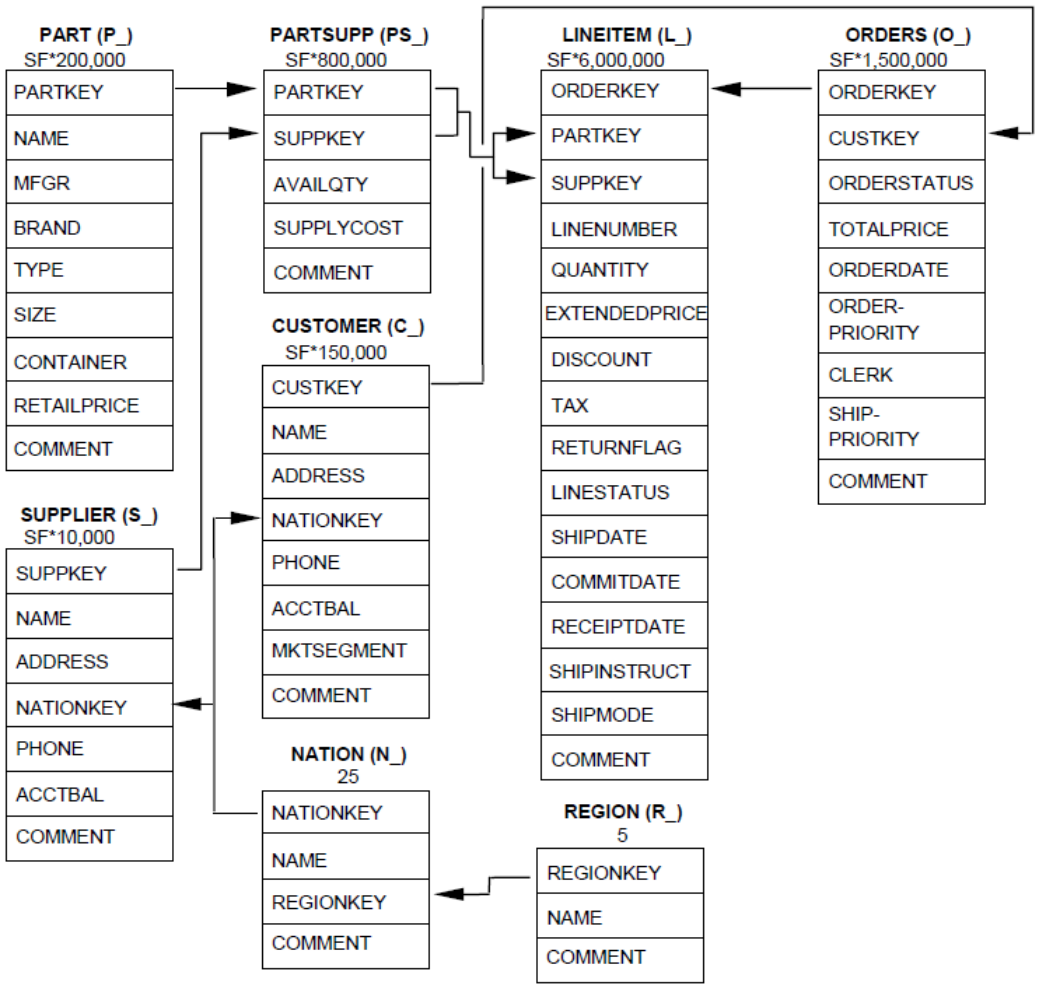


Figura 45 – TPC-H esquema [TPCH, 2008]

3.2 Consultas

Nesta seção, apresentamos as consultas do benchmark TPC-H [TPCH, 2008] que foram utilizadas nos testes experimentais. As consultas foram escolhidas de forma a representar os testes relevantes para avaliação do modelo: inserções, atualizações, deleções, operação merge, consultas simples (ou seja, sem subconsulta, group by etc), consultas com subconsulta, consultas com group by e consultas com having.

C1. Relatório do resumo de preços: Esta consulta lista a quantidade que foi vendida, remetida e retornada (Figura 46).

```
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice*(1-l_discount)) as sum_disc_price,
    sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval '[DELTA]' day (3)
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

Figura 46 – Consulta C1: Relatório do resumo de preços

C2. Fornecedor com menor custo: Esta consulta determina o fornecedor que deverá ser selecionado para atender uma determinada parte em uma determinada região (Figura 47).

```
select
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = [SIZE]
    and p_type like '%[TYPE]'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = '[REGION]'
    and ps_supplycost = (
            select
                min(ps_supplycost)
            from
                partsupp, supplier,
                nation, region
            where
                p_partkey = ps_partkey
                and s_suppkey = ps_suppkey
                and s_nationkey = n_nationkey
                and n_regionkey = r_regionkey
                and r_name = '[REGION]'
)
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey;
```

Figura 47 - Consulta C2: Fornecedor com menor custo

C3. Prioridade de expedição: Esta consulta retorna os 10 pedidos com o valor mais alto que não foram remetidos (Figura 48).

```
select
    l_orderkey,
    sum(l_extendedprice*(1-l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = '[SEGMENT]'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '[DATE]'
    and l_shipdate > date '[DATE]'
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
```

Figura 48 - Consulta C3: Prioridade de expedição

C4. Checagem de prioridade de pedido: Esta consulta determina o quão bem o sistema de priorização de pedidos está funcionando e apresenta uma avaliação da satisfação dos clientes (Figura 49).

```
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '[DATE]'
    and o_orderdate < date '[DATE]' + interval '3' month
    and exists (
            select
                *
            from
                lineitem
            where
                l_orderkey = o_orderkey
                and l_commitdate < l_receiptdate
        )
group by
    o_orderpriority
order by
    o_orderpriority;
```

Figura 49 – Consulta C4: Checagem de prioridade de pedido

C5. Volume do fornecedor local: Esta consulta lista o volume de receita de fornecedores locais (Figura 50).

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = '[REGION]'
    and o_orderdate >= date '[DATE]'
    and o_orderdate < date '[DATE]' + interval '1' year
group by
    n_name
order by
    revenue desc;
```

Figura 50 - Consulta C5: Volume do fornecedor local

4 Exemplos de regras de autorização

Esta seção descreve exemplos de regras de autorização elaborados a partir das consultas do TPC-H. Além disso, as seguintes adaptações foram realizadas no modelo TPC-H:

- Inclusão de campo USERNAME na tabela CUSTOMER.
- Inclusão de tabela USER associada à tabela NATION.
- Inclusão de atributo hemisfério na tabela NATION.
- Inclusão de novas consultas:

C6. tens de pedido de determinados clientes: Esta consulta lista a quantidade, preço, desconto e taxa de envio dos itens de pedidos dos clientes. Essa consulta foi criada para testar a regra que restringe o acesso aos itens de pedido dos clientes que podem ser acessados por um determinado usuário (Figura 51).

```
select distinct
    l_linenumber,
    l_quantity,
    l_extendedprice,
    l_discount,
    l_tax
from
   lineitem;
```

Figura 51 - Consulta C6: Itens de pedido de determinados clientes

C7. Informações de embarque dos itens de pedido: Esta consulta lista as informações de embarque dos itens de todos os pedidos. Esta consulta foi criada para testar a regra que restringe o acesso às informações de embarque (Figura 52).

```
select distinct
    l_linenumber,
    l_shipdate,
    l_shipinstruct,
    l_shipmode
from
   lineitem;
```

Figura 52 - Consulta C7: Informações de embarque dos itens de pedido

As seguintes regras de autorização foram elaboradas:

T1. **Proposta de regra**: Restringir o acesso pelo *Gerente de vendas* aos itens de pedidos provenientes de fornecedores das nações do hemisfério Norte das regiões Ásia e América.

Perfil: Para este exemplo estamos considerando um usuário com perfil de *Gerente de vendas* que tem acesso às informações relativas aos fornecedores do hemisfério Norte (1) das regiões Ásia e América.

T2. **Proposta de regra:** Restringir o acesso às informações de embarque contidos na tabela item de pedido (LINEITEM) para usuários do perfil *Gerente de vendas*.

Perfil: Para este exemplo estamos considerando um usuário com perfil de *Gerente de vendas*, definido no exemplo anterior.

T3. **Proposta de regra**: Restringir acesso à tabela item de pedido (LINEITEM) para clientes (CUSTOMER) quando o acesso for utilizando aplicativo APL1. Somente clientes cadastros na aplicação APL1 e das nações Moçambique ou Índia ou Rússia poderão ter acesso aos seus itens de pedidos.

Perfil: Para esta regra estamos considerando um usuário com o perfil *Cliente* que tem acesso apenas às suas compras.

T4. **Proposta de regra**: Restringir acesso aos dados de fornecedores (SUPPLIERS) para os sistemas regionais, os usuários de uma nação só podem consultar fornecedores associados à sua nação.

Perfil: Para esta regra estamos considerando um usuário com perfil *Gestor de depósito local* que tem acesso apenas as informações dos fornecedores da mesma nação que a dele.

T5. **Proposta de regra**: Permitir que usuários da nação Romênia possam acessar os itens de pedido de clientes das nações Brasil e Argentina.

Perfil: Para esta regra estamos considerando um usuário com perfil Marketing que se localiza na nação Romênia, mas realiza prospecção de produtos para as nações Brasil e Argentina.

5 Avaliação do Label Security

Esta seção apresenta exemplos de uso do Label Security no modelo de dados do TPC-H como descrito na seção 2.

5.1 Regras cadastradas

Esta seção apresenta as regras de autorização criadas para os exemplos de regras apresentados na seção 4.

5.1.1 Exemplo T1

Para o exemplo T1, correspondente à regra de autorização "Restringir o acesso pelo Gerente de vendas aos fornecedores das nações do hemisfério Norte das regiões Ásia e América", foi criada a política chamada GERENCIA que adiciona a coluna GERLAB às tabelas onde a política será aplicada (Figura 53).

```
EXECUTE
SA_SYSDBA.CREATE_POLICY('GERENCIA','GERLAB','READ_CONTROL,CHECK_CONTROL,LABEL_DEFAULT');
```

Figura 53 - Comando de criação da política GERENCIA

Com o intuito da política ser genérica o suficiente para criação de novas políticas, a hierarquia existente entre *Nation* e *Region* foi utilizada para definir níveis para a política. Dessa forma, foi criado um nível de sensibilidade de dado chamado *Nation*, como apresentado na Figura 54. Observe que pelo menos um nível deve ser cadastrado. Em seguida, foram criados diversos grupos relacionados às nações, como apresentado na Figura 55. A escolha por utilizar grupos foi feita porque pode-se definir hierarquias utilizando os mesmos, como se deseja realizar com a hierarquia de nações e regiões e hemisférios, como mostram a Figura 54 e a Figura 55. Por exemplo, o grupo GER_L_ALGERIA refere-se ao grupo que inclui a nação Argélia, enquanto que o grupo GER_AFR_N corresponde ao grupo que inclui as nações da África localizadas no hemisfério norte. Dessa forma, o grupo GER_L_ALGERIA é filho do grupo GER_AFR_N, o que pode ser evidenciado pela coluna *pai* do grupo GER_L_ALGERIA cujo valor é igual a GER_AFR_N. Logo, GER_L_ALGERIA está no primeiro nível da hierarquia, enquanto que GER_AFR_N está no segundo nível da hierarquia. Além disso, criamos o grupo GER_GERAL que é pai de todos os grupos do segundo nível da hierarquia.

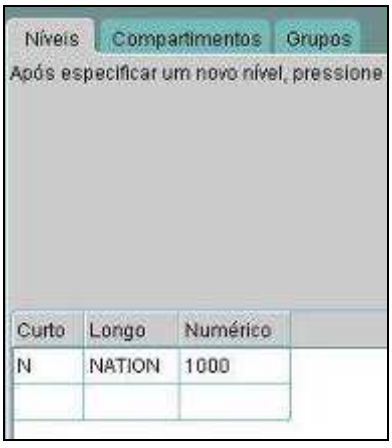


Figura 54 – Níveis da política GERENCIA

Curto	Longo	Numérico	Pai
GER_L_ALGERIA	GER_L_ALGERIA	100	GER_AFR_N
GER_L_ARGENTINA	GER_L_ARGENTINA	101	GER_AM_S
GER_L_BRAZIL	GER_L_BRAZIL	102	GER_AM_S
GER_L_CANADA	GER_L_CANADA	103	GER_AM_N
GER_L_EGYPT	GER_L_EGYPT	104	GER_AFR_N
GER_L_ETHIOPIA	GER_L_ETHIOPIA	105	GER_AFR_N
GER_L_FRANCE	GER_L_FRANCE	106	GER_EU_N
GER_L_GERMANY	GER_L_GERMANY	107	GER_EU_N
GER_L_INDIA	GER_L_INDIA	108	GER_ASIA_N
GER_L_INDONESIA	GER_L_INDONESIA	109	GER_ASIA_S
GER_AFR_N	GER_AFR_N	200	GER_GERAL
GER_AM_N	GER_AM_N	201	GER_GERAL
GER_AFR_S	GER_AFR_S	202	GER_GERAL
GER_AM_S	GER_AM_S	203	GER_GERAL
GER_ASIA_N	GER_ASIA_N	204	GER_GERAL
GER_ASIA_S	GER_ASIA_S	205	GER_GERAL
GER_EU_N	GER_EU_N	206	GER_GERAL
GER_ME_N	GER_ME_N	207	GER_GERAL
GER_ASIA_NS	GER_ASIA_NS	208	GER_GERAL
GER_GERAL	GER_GERAL	300	
GER_L_IRAN	GER_L_IRAN	1010	GER_ME_N
GER_L_IRAQ	GER_L_IRAQ	1011	GER_ME_N
GER_L_JAPAN	GER_L_JAPAN	1012	GER_ASIA_N
GER_L_JORDAN	GER_L_JORDAN	1013	GER_ME_N

Figura 55 – Grupos da política GERENCIA

Após a criação dos componentes dos rótulos, são criados os rótulos para os dados. Neste exemplo, foram criados os rótulos para as nações, todos para o nível NATION e para os grupos correspondentes no primeiro nível da hierarquia. Isto porque existe apenas o nível NATION, não existe nenhum compartimento e as linhas são rotuladas pelo grupo mais específico, no caso, a nação. Se o usuário tem acesso a um grupo do nível mais alto da hierarquia, então ele tem acesso aos subgrupos desse grupo (ou seja, do nível mais baixo). Exemplos de rótulos criados são apresentados na Figura 56. Como exemplo, foi criado o rótulo N::GER_L_ALGERIA, que significa rótulo para o nível N (NATION) e para o subgrupo (GER_L_ALGERIA). O número de tag 1100

representa o rótulo que será atribuído à coluna GERLAB nos registros da tabela para cada regra definida.

Figura 56 – Rótulos de dados

Essa política foi aplicada tanto na tabela BASE_SUPPLIER quanto na tabela BASE_LINEITEM, restringindo o acesso do Gerente de Vendas para os fornecedores e itens fornecidos pelos fornecedores.

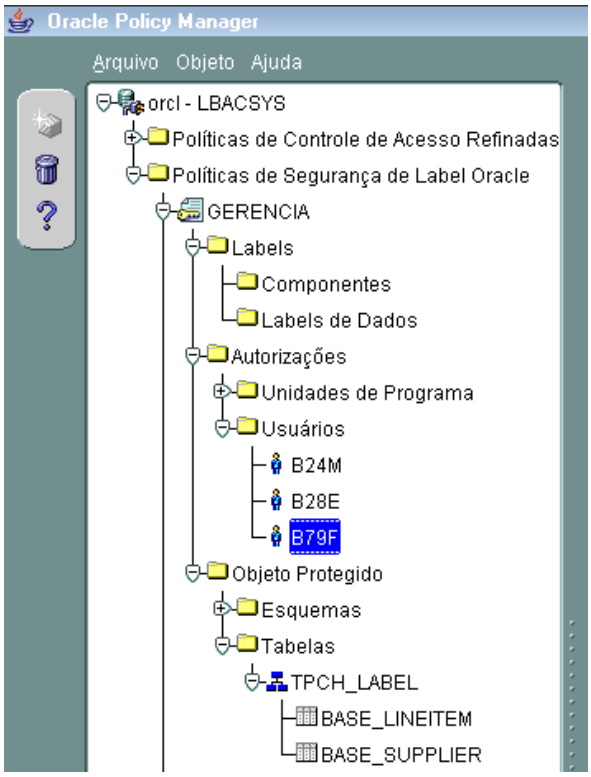


Figura 57 – Estrutura da política GERENCIA

5.1.1.1 Teste de consulta

Para o teste de consulta, foram utilizados os usuários B24M e lbacsys. Ao B24M foi atribuído acesso aos grupos "GER_L_FRANCE" com a tag "1106" para "N::GER_L_FRANCE". O usuário lbacsys tem o privilégio EXEMPT POLICY, logo as regras de autorização não são aplicadas a ele. Utilizando o usuário B24M foi feita uma consulta de função agregada *count(*)* retornando 67660 registros, como apresentado na Figura 58. Utilizando o usuário lbacsys são retornados 1793873 registros.

```
select count(*) from tpch_label.base_lineitem;
```

Figura 58 - Comando de consulta na tabela BASE_LINEITEM

	COUNT(*)
1	67660

Figura 59 - Select na tabela BASE_LINEITEM com usuário B24M

	COUNT(*)
1	1793873

Figura 60 - Select na tabela BASE_LINEITEM com usuário lbacsys

5.1.2 Exemplo T2

Para o exemplo T2, a restrição "*Gerente de vendas* não poderá ter acesso às informações de embarque dos itens de pedido"não pode ser representada pelo Label Security, visto que ele restringe acesso por linhas e não suporta restrição por colunas.

5.1.3 Exemplo T3

Para o exemplo T3, referente à restrição "Restringir acesso à tabela item de pedido (LINEITEM) para clientes (CUSTOMER) quando o acesso for utilizando aplicativo APL1. Somente clientes cadastros na aplicação APL1 e das nações Moçambique ou Índia ou Rússia poderão ter acesso aos seus pedidos" foi criada uma nova política que atribui rótulos de acordo com as nações para as tabelas BASE_ORDERS e BASE_LINEITEM, como feito na política GERENCIA. Aparentemente poderia-se pensar em utilizar a mesma política GERENCIA para os dois casos, pois esta permite restrição por nação. No entanto, as políticas têm objetivos diferentes, um mesmo usuário com autorização para os pedidos de Moçambique, pode não ter autorização para os fornecedores de Moçambique, por exemplo, e utilizando a mesma política, ocorreriam conflitos.

Logo, para o exemplo T3 foi criada a política RESTRINGE_PEDIDO_CLIENTE que cria um rótulo de dados para os pedidos de cliente de cada nação (Índia, Moçambique e Rússia) e mais um para os pedidos de clientes das outras nações. O nome da coluna criada foi RESTPEDCLI, e os rótulos são "P::MO" para Moçambique, "P::ID" para Índia, "P::RU" para Rússia e "P::XX" para as demais nações. Mais uma vez, foi utilizado apenas um nível de sensibilidade nomeado PUBLICO e os grupos referentes a MO, ID, RU, XX e TO (todas as nações), Figura 61 e Figura 62.

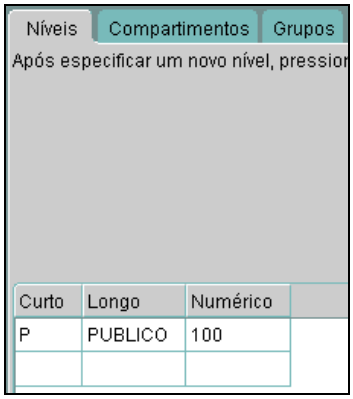


Figura 61 – Níveis da política RESTRINGE_PEDIDO_CLIENTE

Figura 62 – Grupos da política RESTRINGE_PEDIDO_CLIENTE

Exemplos de rótulos criados são apresentados na Figura 63. Como exemplo, foi criado o rótulo P::ID, que significa rótulo para o nível P (PUBLICO) e para o subgrupo (ID, referente à Índia). O número de tag '201100 representa o rótulo a ser atribuído nos registros na coluna RESTPEDCLI nos registros da tabela.

Figura 63 – Rótulos de dados

Neste teste prático, o usuário B24M foi autorizado como cliente de Moçambique, podendo acessar apenas as linhas relacionadas à nação Moçambique, recebendo o rótulo "P::MO". Essa configuração é ilustrada na Figura 64 e Figura 65

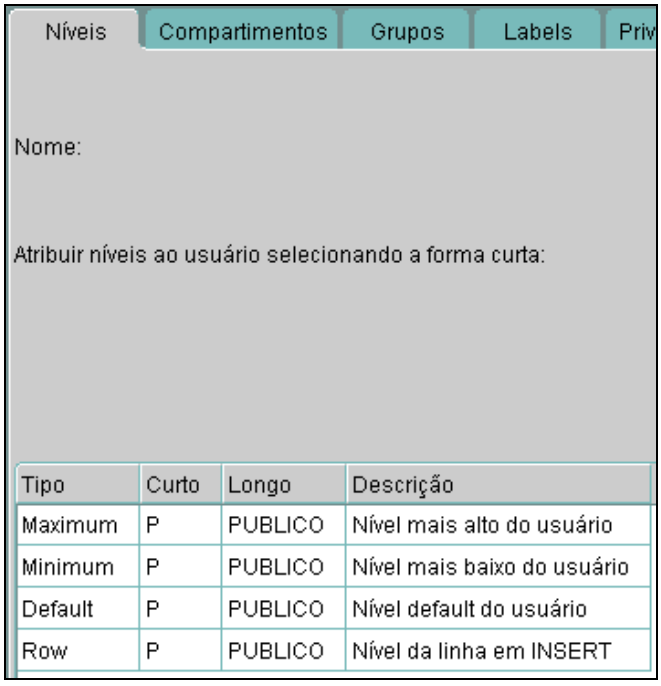


Figura 64 – Níveis atribuídos a B24M

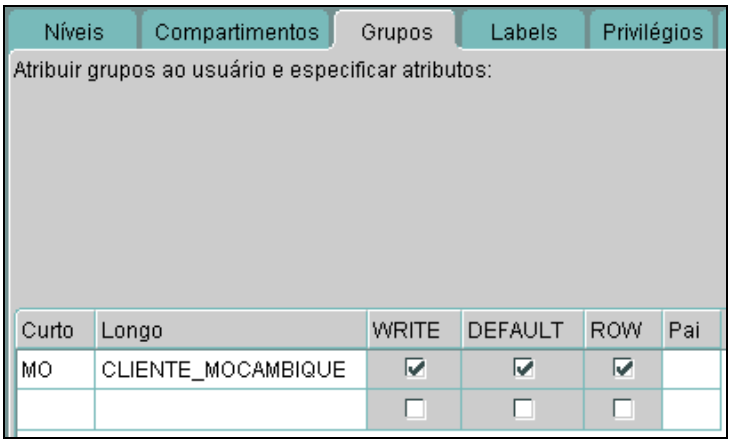


Figura 65 – Grupos atribuídos a B24M

Essa política foi aplicada tanto na tabela BASE_ORDERS quanto na tabela BASE_LINEITEM, restringindo o acesso dos clientes e itens dos pedidos dos clientes.

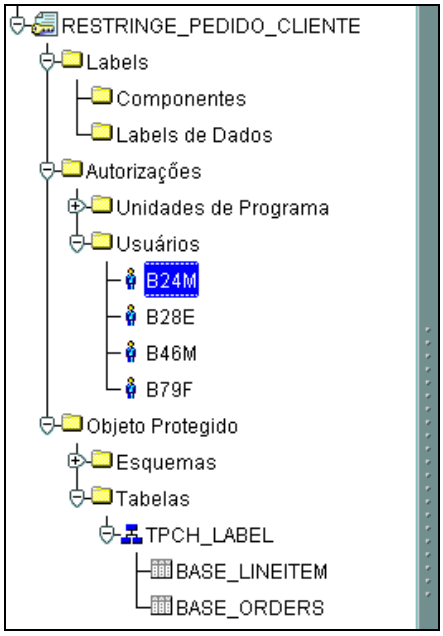


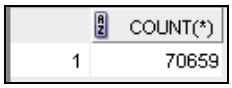
Figura 66 – Estrutura da política RESTRINGE_PEDIDO_CLIENTE

5.1.3.1 Teste de consulta

Para o teste de consulta, foram utilizados os usuários B24M (rótulo "P::ID" com acesso ao rótulo de tag "201100") e lbacsys. O usuário lbacsys tem o privilégio EXEMPT POLICY, logo as regras de autorização não são aplicadas a ele. Utilizando o usuário B24M foi feita uma consulta de função agregada *count(*)* retornando 70659 registros, como apresentado na Figura 58. Utilizando o usuário lbacsys são retornados 1793873 registros.

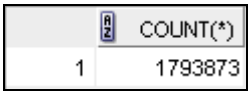
```
select count(*) from tpch_label.base_lineitem;
```

Figura 67 - Comando de consulta na tabela BASE_LINEITEM



	COUNT(*)
1	70659

Figura 68 - Select na tabela BASE_LINEITEM com usuário B24M



	COUNT(*)
1	1793873

Figura 69 - Select na tabela BASE_LINEITEM com usuário lbacsys

5.1.3.2 Conflito entre as políticas dos exemplos T1 e T2

As políticas GERENCIA (Exemplo T1) e RESTRINGE_PEDIDO_CLIENTE quando utilizadas em uma mesma tabela podem levar a conflito. Por exemplo, se um usuário é *Gerente de vendas da América e Ásia do hemisfério Norte* e *Cliente de Moçambique* e ambas as políticas foram aplicadas à tabela LINEITEM, o usuário teria acesso aos itens de pedido referentes às nações da América e Ásia que se localizam no hemisfério Norte pela política GERENCIA. Além disso, por esta política, ele não teria acesso a nenhuma outra nação além destas. Já pela política RESTRINGE_PEDIDO_CLIENTE, ele teria acesso apenas aos pedidos que foram feitos para a nação Moçambique e nenhuma outra. Como a interseção do resultado da primeira política com a segunda política retorna vazio, então o usuário não terá acesso a nenhum item de pedido.

Para o teste, mantivemos a autorização do usuário B79F com o rótulo "N::GER_AM_N,GER_ASIA_N,GER_ASIA_NS" na política GERENCIA e autorizamos o mesmo usuário na política RESTRINGE_PEDIDO_CLIENTE com o rótulo "P::MO". As duas políticas ativadas impedem o usuário de acessar qualquer registro da tabela BASE_LINEITEM, pois, por uma, ele pode apenas visualizar registros de fornecedores das nações da América e Ásia que se localizam no hemisfério Norte e pelo outro ele pode apenas visualizar registros de clientes de Moçambique. As políticas funcionam separadamente, quando desativa uma política ou outra.

5.1.4 Exemplo T4

Para o exemplo T4, a restrição "Restringir acesso aos dados de fornecedores (SUPPLIERS) para os sistemas regionais, os usuários de uma nação só podem consultar fornecedores associados à sua nação" é importante ressaltar que não é possível criar uma política no Label Security baseada no valor de uma coluna de outra tabela. O label security permite a criação de rótulos fixos e não de rótulos dinâmicos. Dessa forma, foi criada a política RESTRINGE_CLI_FORN que autoriza o usuário a acessar um grupo, criando um rótulo de dados para os fornecedores de cada nação. Não é possível associar a nação definida para o grupo à nação do usuário. Logo, o grupo será atribuído manualmente ao usuário de acordo com o valor do atributo n_nationkey do usuário. Se o valor do atributo for alterado, a atribuição do usuário ao grupo não é automaticamente alterada, sendo necessário realizar esta alteração manualmente. O nome da coluna criada foi RESTCLIFORN, e a configuração da política junto com a autorização de um usuário com acesso aos fornecedores da China são apresentadas na Figura 61

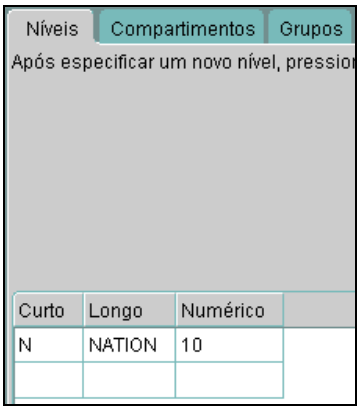
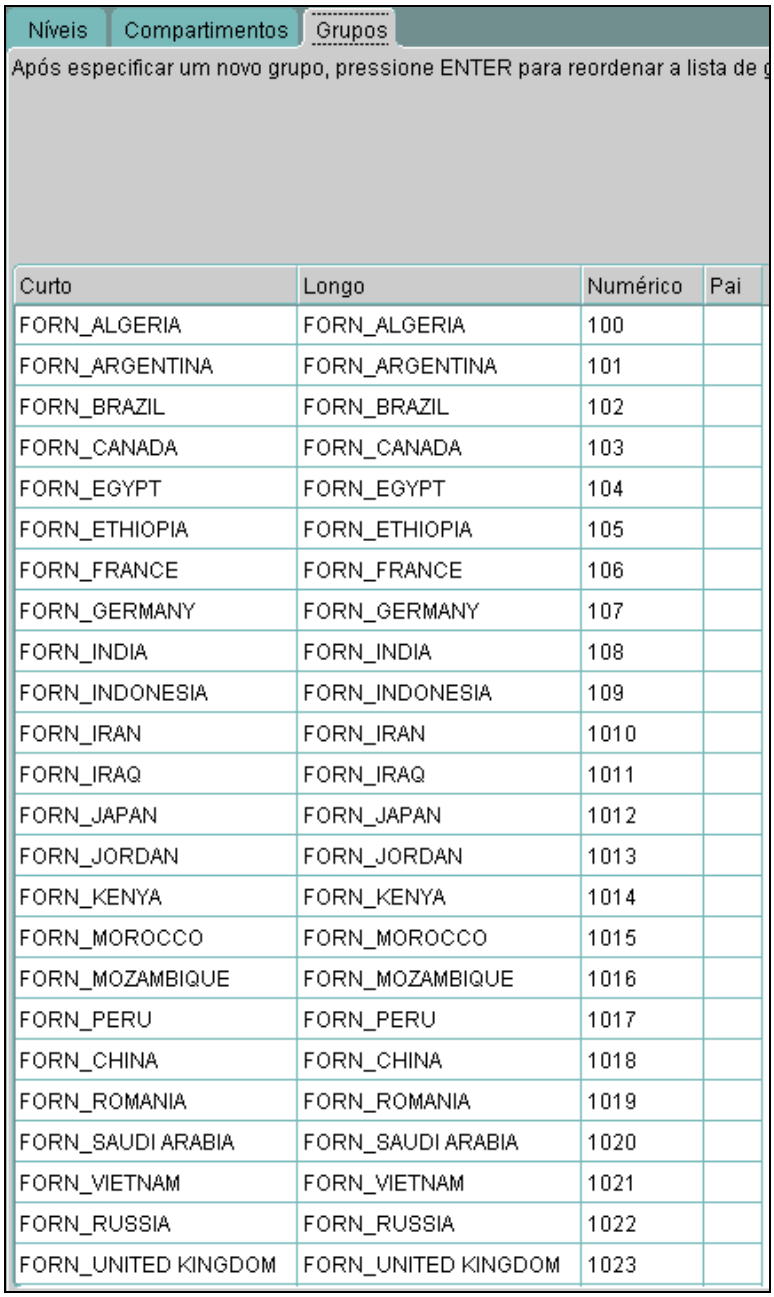
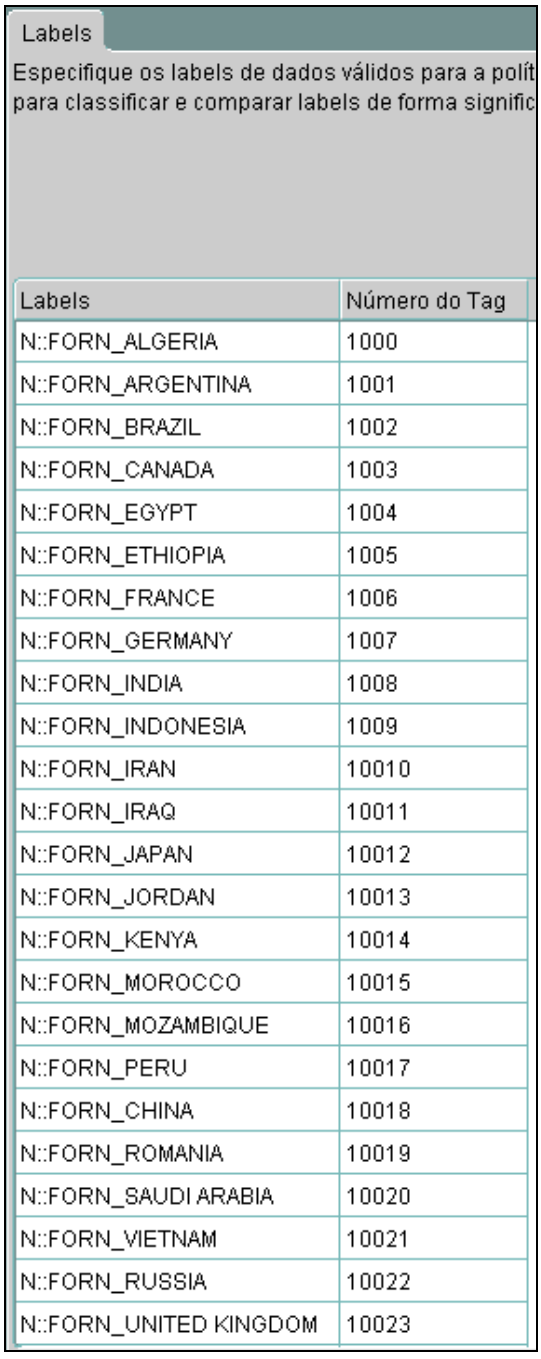


Figura 70 – Níveis da política RESTRINGE_CLI_FORN



Curto	Longo	Numérico	Pai
FORN_ALGERIA	FORN_ALGERIA	100	
FORN_ARGENTINA	FORN_ARGENTINA	101	
FORN_BRAZIL	FORN_BRAZIL	102	
FORN_CANADA	FORN_CANADA	103	
FORN_EGYPT	FORN_EGYPT	104	
FORN_ETHIOPIA	FORN_ETHIOPIA	105	
FORN_FRANCE	FORN_FRANCE	106	
FORN_GERMANY	FORN_GERMANY	107	
FORN_INDIA	FORN_INDIA	108	
FORN_INDONESIA	FORN_INDONESIA	109	
FORN_IRAN	FORN_IRAN	1010	
FORN_IRAQ	FORN_IRAQ	1011	
FORN_JAPAN	FORN_JAPAN	1012	
FORN_JORDAN	FORN_JORDAN	1013	
FORN_KENYA	FORN_KENYA	1014	
FORN_MOROCCO	FORN_MOROCCO	1015	
FORN_MOZAMBIQUE	FORN_MOZAMBIQUE	1016	
FORN_PERU	FORN_PERU	1017	
FORN_CHINA	FORN_CHINA	1018	
FORN_ROMANIA	FORN_ROMANIA	1019	
FORN_SAUDI ARABIA	FORN_SAUDI ARABIA	1020	
FORN_VIETNAM	FORN_VIETNAM	1021	
FORN_RUSSIA	FORN_RUSSIA	1022	
FORN_UNITED KINGDOM	FORN_UNITED KINGDOM	1023	

Figura 71 – Grupos da política RESTRINGE_CLI_FORN



Labels	Número do Tag
N::FORN_ALGERIA	1000
N::FORN_ARGENTINA	1001
N::FORN_BRAZIL	1002
N::FORN_CANADA	1003
N::FORN_EGYPT	1004
N::FORN_ETHIOPIA	1005
N::FORN_FRANCE	1006
N::FORN_GERMANY	1007
N::FORN_INDIA	1008
N::FORN_INDONESIA	1009
N::FORN_IRAN	10010
N::FORN_IRAQ	10011
N::FORN_JAPAN	10012
N::FORN_JORDAN	10013
N::FORN_KENYA	10014
N::FORN_MOROCCO	10015
N::FORN_MOZAMBIQUE	10016
N::FORN_PERU	10017
N::FORN_CHINA	10018
N::FORN_ROMANIA	10019
N::FORN_SAUDI ARABIA	10020
N::FORN_VIETNAM	10021
N::FORN_RUSSIA	10022
N::FORN_UNITED KINGDOM	10023

Figura 72 – Rótulos de dados

Figura 73 – Níveis atribuídos a B24M

Figura 74 – Grupos atribuídos a B24M

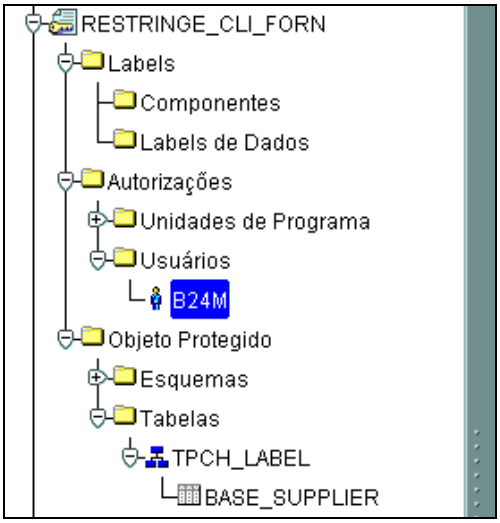


Figura 75 – Estrutura da política RESTRINGE_CLI_FORN

5.1.4.1 Teste de consulta

Para o teste de consulta, foram utilizados os usuários B24M (rótulo "N::FORN_CHINA" com acesso ao rótulo de tag "10018") e lbacsys. O usuário lbacsys tem o privilégio EXEMPT POLICY, logo as regras de autorização não são aplicadas a ele. Utilizando o usuário B24M foi feita uma consulta de função agregada *count(*)* retornando 144 registros, como apresentado na Figura 58. Utilizando o usuário lbacsys são retornados 2990 registros.

```
select count(*) from tpch_label.base_supplier;
```

Figura 76 - Comando de consulta na tabela BASE_ SUPPLIER

	COUNT(*)
1	144

Figura 77 - Select na tabela BASE_ SUPPLIER com usuário B24M

	COUNT(*)
1	2990

Figura 78 - Select na tabela BASE_SUPPLIER com usuário lbacsys

5.1.5 Exemplo T5

Para o exemplo T5, a restrição "Permitir que usuários da nação Romênia possam acessar os itens de pedido de clientes das nações Brasil e Argentina" foi criada a política RESTRINGE_ITENS_BRA_ROM que autoriza o usuário a acessar os grupos Brasil e Argentina, criando um rótulo de dados para os fornecedores de cada nação. O nome da coluna criada foi RESTITBRAROM, e a configuração da política junto com a autorização de um usuário com acesso aos itens de pedido do Brasil e da Argentina são apresentados na Figura 79, Figura 80, Figura 81, Figura 82, Figura 83 e Figura 84.

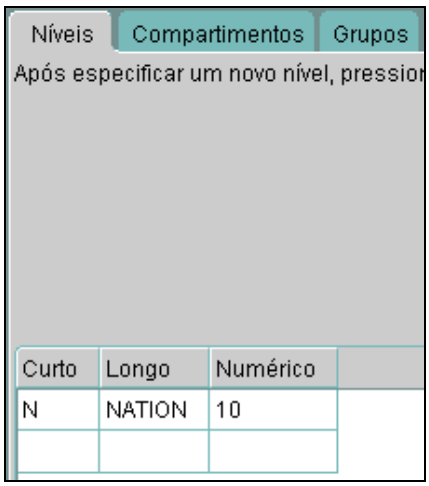
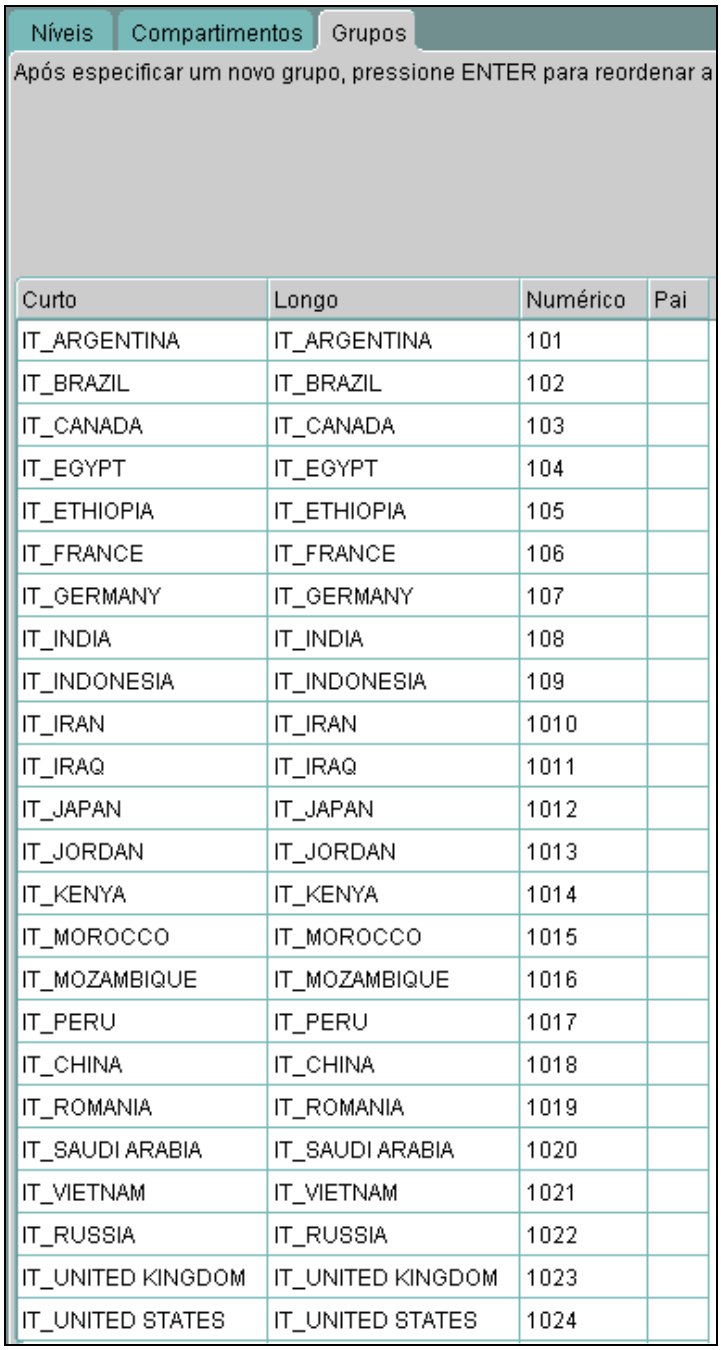
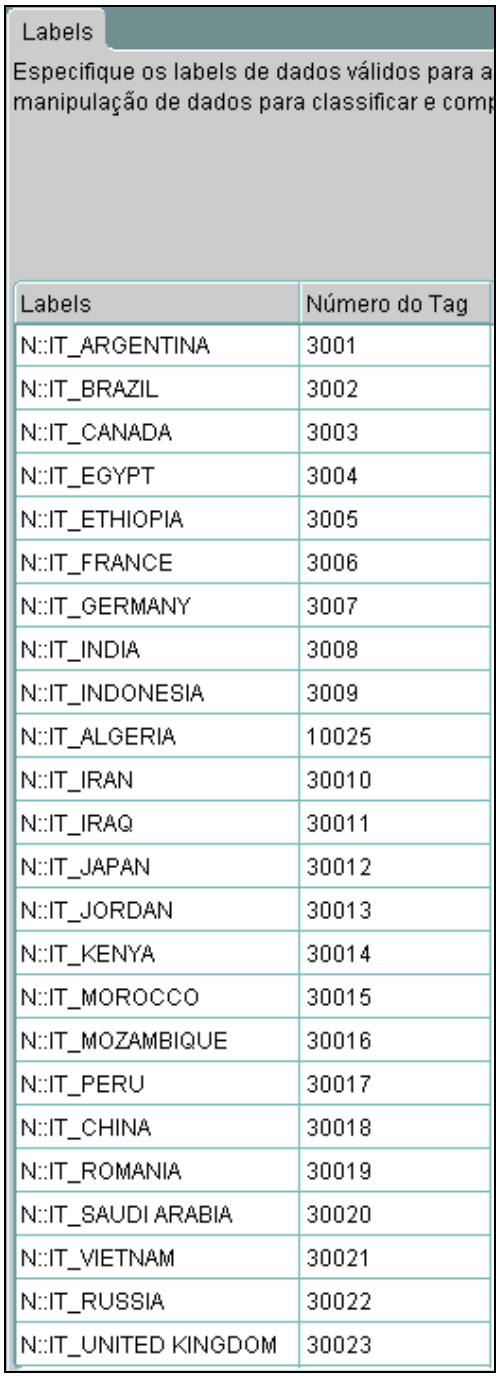


Figura 79 – Níveis da política RESTRINGE_ITENS_BRA_ROM



Curto	Longo	Numérico	Pai
IT_ARGENTINA	IT_ARGENTINA	101	
IT_BRAZIL	IT_BRAZIL	102	
IT_CANADA	IT_CANADA	103	
IT_EGYPT	IT_EGYPT	104	
IT_ETHIOPIA	IT_ETHIOPIA	105	
IT_FRANCE	IT_FRANCE	106	
IT_GERMANY	IT_GERMANY	107	
IT_INDIA	IT_INDIA	108	
IT_INDONESIA	IT_INDONESIA	109	
IT_IRAN	IT_IRAN	1010	
IT_IRAQ	IT_IRAQ	1011	
IT_JAPAN	IT_JAPAN	1012	
IT_JORDAN	IT_JORDAN	1013	
IT_KENYA	IT_KENYA	1014	
IT_MOROCCO	IT_MOROCCO	1015	
IT_MOZAMBIQUE	IT_MOZAMBIQUE	1016	
IT_PERU	IT_PERU	1017	
IT_CHINA	IT_CHINA	1018	
IT_ROMANIA	IT_ROMANIA	1019	
IT_SAUDI ARABIA	IT_SAUDI ARABIA	1020	
IT_VIETNAM	IT_VIETNAM	1021	
IT_RUSSIA	IT_RUSSIA	1022	
IT_UNITED KINGDOM	IT_UNITED KINGDOM	1023	
IT_UNITED STATES	IT_UNITED STATES	1024	

Figura 80 – Grupos da política RESTRINGE_ITENS_BRA_ROM



Labels	Número do Tag
N::IT_ARGENTINA	3001
N::IT_BRAZIL	3002
N::IT_CANADA	3003
N::IT_EGYPT	3004
N::IT_ETHIOPIA	3005
N::IT_FRANCE	3006
N::IT_GERMANY	3007
N::IT_INDIA	3008
N::IT_INDONESIA	3009
N::IT_ALGERIA	10025
N::IT_IRAN	30010
N::IT_IRAQ	30011
N::IT_JAPAN	30012
N::IT_JORDAN	30013
N::IT_KENYA	30014
N::IT_MOROCCO	30015
N::IT_MOZAMBIQUE	30016
N::IT_PERU	30017
N::IT_CHINA	30018
N::IT_ROMANIA	30019
N::IT_SAUDI ARABIA	30020
N::IT_VIETNAM	30021
N::IT_RUSSIA	30022
N::IT_UNITED KINGDOM	30023

Figura 81 – Rótulos de dados

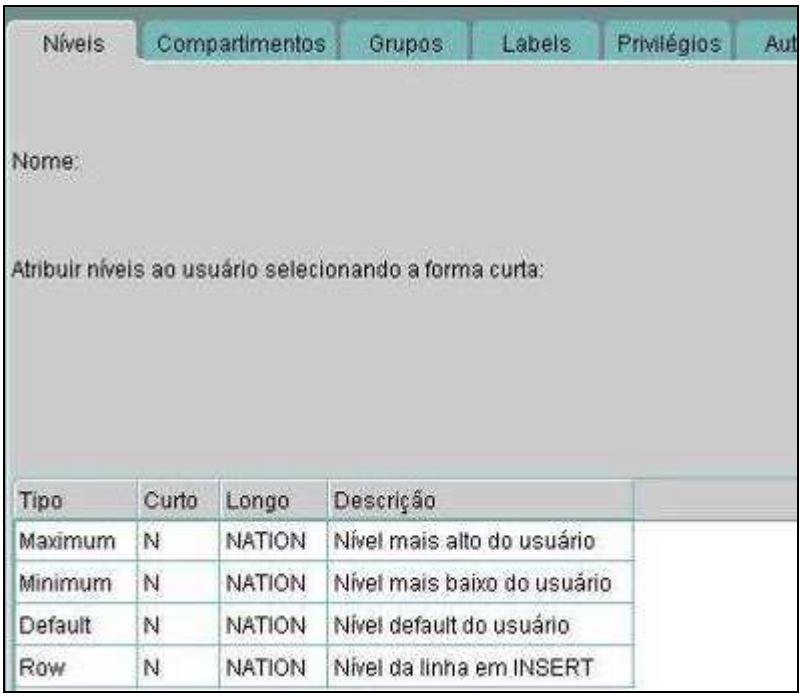


Figura 82 – Níveis atribuídos a B24M

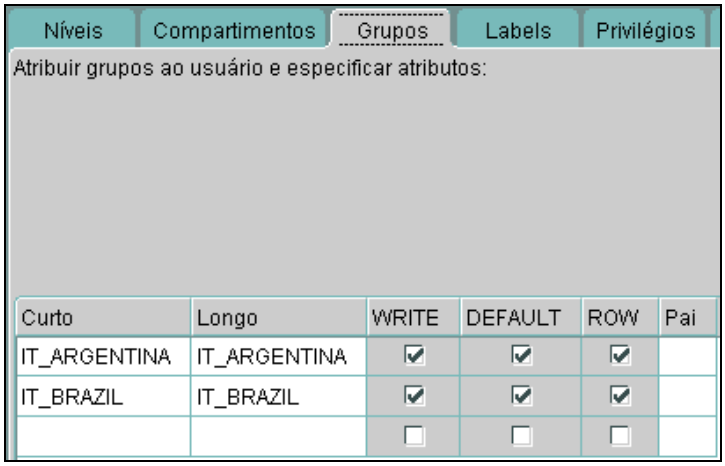


Figura 83 – Grupos atribuídos a B24M

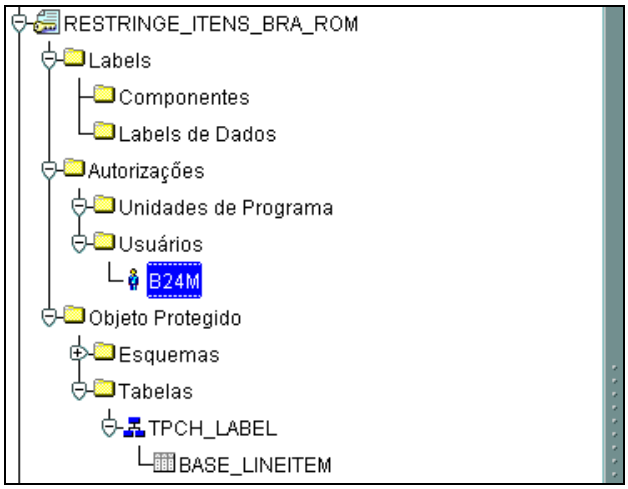


Figura 84 – Estrutura da política RESTRINGE_ITENS_BRA_ROM

5.1.5.1 Teste de consulta

Para o teste de consulta, foram utilizados os usuários B24M (rótulo "N::IT_ARGENTINA,IT_BRAZIL" com acesso aos rótulos de tag "3001 " e "3002") e lbacsys. O usuário lbacsys tem o privilégio EXEMPT POLICY, logo as regras de autorização não são aplicadas a ele. Utilizando o usuário B24M foi feita uma consulta de função agregada *count(*)* retornando 145863 registros, como apresentado na Figura 58. Utilizando o usuário lbacsys são retornados 1793873 registros.

```
select count(*) from tpch_label.base_lineitem;
```

Figura 85 - Comando de consulta na tabela BASE_LINEITEM

	COUNT(*)
1	145863

Figura 86 - Select na tabela BASE_LINEITEM com usuário B24M

	COUNT(*)
1	1793873

Figura 87 - Select na tabela BASE_LINEITEM com usuário lbacsys

5.2 Testes de escrita

Para os testes de escrita, que envolvem inserção, atualização e remoção, foram utilizados os usuários B24M e B28E. A eles foi atribuída a política "GERENCIA" com acesso aos grupos "GER_L_FRANCE" e "GER_L_UNITED KINGDOM", respectivamente. As *tags* (número que aparece na coluna da tabela onde a política foi aplicada) referentes aos rótulos de dados são "1106" para "N::GER_L_FRANCE" e "11023" para "N::GER_L_UNITED KINGDOM". Em seguida, usando o usuário B24M foi criado um registro com chave primária 59801 na tabela BASE_PART para ser usado como referência nos testes. Depois foi feito um select para visualizar o registro, como apresentado na Figura 88 e Figura 89.

```
insert into tpch_label.base_part (p_partkey ,p_name, p_mfgr, p_brand,
p_type, p_size, p_container, p_retailprice, p_comment) values (59801,
'xx', 'Manufacturer#8', 'Brand#8', 'PROMO BURNISHED COPPER', 1, 'LG
CASE', '800', 'xxxxxx');


select * from tpch_label.base_part where p_partkey = 59801;
```

Figura 88 - Comando de inserção na tabela BASE_PART

	P_PARTKEY	P_NAME	P_MFGR	P_BRAND	P_TYPE	P_SIZE	P_CONTAINER	P_RETAILPRICE	P_COMMENT
1	59801	xx	Manufacturer#8	Brand#8	PROMO BURNISHED COPPER	1	LG CASE	800	xxxxxx

Figura 89 – Select na tabela BASE_PART com P_PARTKEY = 59801

5.2.1 Inserção

No teste de inserção, foi utilizado o usuário B24M para criar um registro na tabela BASE_LINEITEM com o L_PARTKEY igual ao P_PARTKEY do registro inserido anteriormente em BASE_PART. O rótulo de dados da política é criado

automaticamente, de acordo com a autorização do usuário B24M na tabela, neste caso foi atribuído o rótulo ”1106” (Figura 90 e Figura 91).

```
insert into tpch_label.base_lineitem (l_orderkey, l_partkey,
l_suppkey, l_linenumber, l_quantity, l_extendedprice, l_discount,
l_tax, l_returnflag, l_linestatus, l_shipdate, l_commitdate,
l_receiptdate, l_shipinstruct, l_shipmode, l_comment) values (393062,
59801, 1502, 3, 25, 22675, '0,01', '0,08', 'R', 'F', '18/04/93',
'27/04/93', '26/04/93', 'DELIVER IN PERSON', 'TRUCK', 'xxxxxx');


select * from tpch_label.base_lineitem where l_partkey = 59801;
```

Figura 90 - Comando de inserção na tabela BASE_LINEITEM

	L_ORD...	L_PA...	L_...	...	...	L_EX...	L_...	L...	...	...	L_...	L_C...	L_...	L_SHIPINSTRUCT	L_S...	L_COMMENT	GERLAB
1	393062	59801	1502	3	25	22675	0,01	0,08	R	F	18/04/93	27/04/93	26/04/93	DELIVER IN PERSON	...	TRUCK	xxxxxx	1106

Figura 91 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

Se B24M tentar inserir um registro com um rótulo que ele não tem acesso (no teste o “1100” para Argélia), aparece uma mensagem de erro, pois B24M tem acesso aos grupos “GER_L_FRANCE” e “GER_L_UNITED KINGDOM”, os quais não incluem a Argélia.

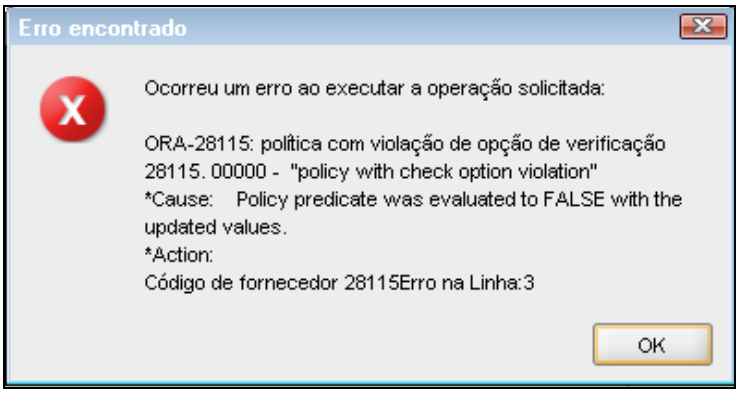


Figura 92 – Mensagem de erro

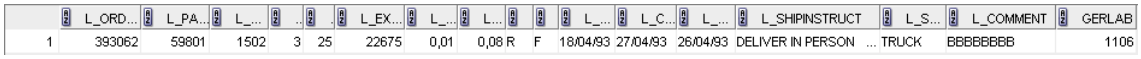
Outro teste realizado foi utilizando o usuário B79F da política GERENCIA (5.1.1) cujo rótulo de usuário é N::GER_AM_N,GER_ASIA_N,GER_ASIA_NS. B79F só pode inserir especificando um rótulo na inserção, visto que ele não possui nenhum grupo relativo a rótulo de dados como padrão para inserção. Desabilitando as outras políticas para não criar conflito, como definimos a granularidade mais baixa para rótulos de dados (apenas nações) e B79F tem acesso a grupos mais altos na hierarquia (regiões e hemisférios), os rótulos devem ser especificados na inserção nesses casos.

5.2.2 Atualização

Para o teste de atualização, foram utilizados os usuários B24M e B28E para atualizar a coluna L_COMMENT do registro inserido na inserção. A DML é apresentada na Figura 93. Utilizando o usuário B24M, o qual tem acesso ao registro, a coluna foi alterada, como apresentado na Figura 94. No entanto, utilizando o usuário B28E com outro valor para L_COMMENT a coluna não é alterada, pois este usuário não tem autorização para acessar essa linha e, portanto, não consegue atualizá-la. A execução da regra não retorna erro. Ao fazer um novo select com o usuário B24M, nada foi alterado.

```
UPDATE tpch_label.base_lineitem set l_comment = 'BBBBBBBB' where
l_orderkey = 393062 and l_partkey = 59801 and l_suppkey = 1502;


select * from tpch_label.base_lineitem where l_partkey = 59801;
```

Figura 93 - Comando de atualização na tabela BASE_LINEITEM



	L_ORD...	L_PA...	L_...	..	.	L_EX...	L_...	L...			L_...	L_C...	L_...	L_SHIPINSTRUCT	L_S...	L_COMMENT	GERLAB
1	393062	59801	1502	3	25	22675	0,01	0,08	R	F	18/04/93	27/04/93	26/04/93	DELIVER IN PERSON	... TRUCK	BBBBBBBB	1106

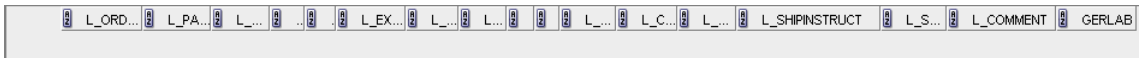
Figura 94 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

5.2.3 Remoção

Para o teste de remoção, foram utilizados os usuários B24M e B28E. Primeiro foi executada a remoção com o usuário B24M (Figura 95). O registro foi removido. Em seguida, o registro foi inserido novamente e executou-se a remoção utilizando o usuário B28E o qual não tem autorização para acessar essa linha. Logo, a remoção não foi realizada. Vale ressaltar que não foi retornado erro ou mensagem indicativa de que a remoção não ocorreu devido a regra de autorização não permitir que o usuário acesse o registro.

```
DELETE tpch_label.base_lineitem where l_orderkey = 393062 and
l_partkey = 59801 and l_suppkey = 1502;


select * from tpch_label.base_lineitem where l_partkey = 59801;
```

Figura 95 - Comando de remoção na tabela BASE_LINEITEM



L_ORD...	L_PA...	L_...	..	.	L_EX...	L_...	L...			L_...	L_C...	L_...	L_SHIPINSTRUCT	L_S...	L_COMMENT	GERLAB

Figura 96 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

5.3 Teste de trigger

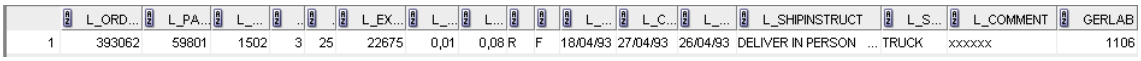
Para o teste de trigger, foi criada uma trigger que atualiza o campo L_COMMENT na tabela BASE_LINEITEM ao alterar o campo P_COMMENT na tabela BASE_PART (Figura 97). Ou seja, se o comentário da part foi alterado, então todos itens de pedido relacionados àquela parte são atualizados.

```
create or replace TRIGGER TRIGGER_TESTE
after UPDATE OF P_COMMENT ON BASE_PART
for each row
BEGIN
  update base_lineitem set l_comment = :NEW.P_COMMENT WHERE l_partkey
= :NEW.P_PARTKEY;
END;
```

Figura 97 – Trigger para teste

O teste consiste em tentar alterar, utilizando a *trigger*, um registro que o usuário B28E não tem permissão para alterar. Em outras palavras, o teste consiste em executar a atualização com o usuário B28E do campo P_COMMENT da tabela BASE_PART e,

verificar se, com a utilização da *trigger* se o campo L_COMMENT dos registros em BASE_LINEITEM são alterados. O registro para o qual tenta-se realizar a alteração é apresentado na Figura 98. A Figura 99 apresenta a DML executada utilizando o usuário B28E. As Figura 100 e Figura 101 apresentam o resultado da execução da atualização. Observe que o *update* atualizou a tabela BASE_PART, pois não há restrição cadastrada para esta tabela. No entanto, a tabela BASE_LINEITEM não foi alterada pela *trigger*, o que é observado comparando as Figura 98 e Figura 101.

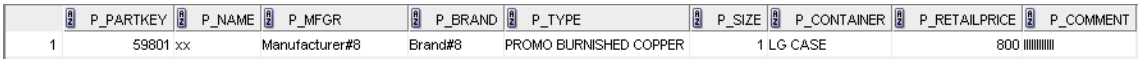


	L_ORD...	L_PA...	L_...	...	...	L_EX...	L_...	L...			L_...	L_C...	L_...	L_SHIPINSTRUCT	L_S...	L_COMMENT	GERLAB
1	393062	59801	1502	3	25	22675	0,01	0,08	R	F	18/04/93	27/04/93	26/04/93	DELIVER IN PERSON	...	TRUCK	xxxxxx	1106

Figura 98 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

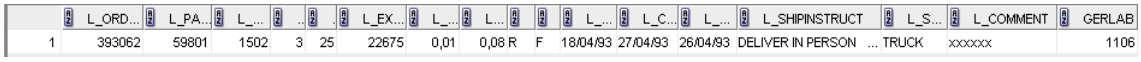
```
update tpch_label.base_part set p_comment = 'IIIIIIIIIII' where
p_partkey = 59801;


select * from tpch_label.base_part where p_partkey = 59801;
```

Figura 99 - Comando de atualização na tabela BASE_PART



	P_PARTKEY	P_NAME	P_MFGR	P_BRAND	P_TYPE	P_SIZE	P_CONTAINER	P_RETAILPRICE	P_COMMENT
1	59801	xx	Manufacturer#8	Brand#8	PROMO BURNISHED COPPER	1	LG CASE	800	IIIIIIIIII

Figura 100 - Select na tabela BASE_PART com P_PARTKEY = 59801



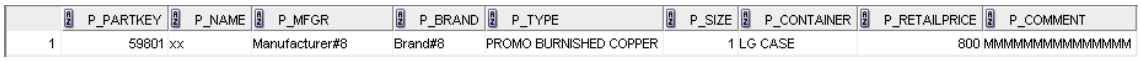
	L_ORD...	L_PA...	L_...	...	...	L_EX...	L_...	L...			L_...	L_C...	L_...	L_SHIPINSTRUCT	L_S...	L_COMMENT	GERLAB
1	393062	59801	1502	3	25	22675	0,01	0,08	R	F	18/04/93	27/04/93	26/04/93	DELIVER IN PERSON	...	TRUCK	xxxxxx	1106

Figura 101 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

Quando B24M faz a mesma atualização em BASE_PART, a trigger executa a modificação no registro em BASE_LINEITEM, pois é o registro que ele tem acesso. A DML é apresentada na Figura 102 e os registros atualizados das tabelas BASE_PART e BASE_LINEITEM são apresentados nas figuras Figura 103 e Figura 104.

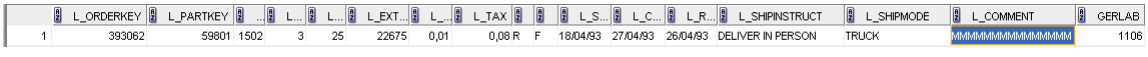
```
update tpch_label.base_part set p_comment = 'MMMMMMMMMMMMMM' where
p_partkey = 59801;


select * from tpch_label.base_part where p_partkey = 59801;
```

Figura 102 - Comando de atualização na tabela BASE_PART



	P_PARTKEY	P_NAME	P_MFGR	P_BRAND	P_TYPE	P_SIZE	P_CONTAINER	P_RETAILPRICE	P_COMMENT
1	59801	xx	Manufacturer#8	Brand#8	PROMO BURNISHED COPPER	1	LG CASE	800	MMMMMMMMMMMMMM

Figura 103 - Select na tabela BASE_PART com P_PARTKEY = 59801



	L_ORDERKEY	L_PARTKEY	...	L...	L...	L_EXT...	L_...	L_TAX			L_S...	L_C...	L_R...	L_SHIPINSTRUCT	L_SHIPMODE	L_COMMENT	GERLAB
1	393062	59801	1502	3	25	22675	0,01	0,08	R	F	18/04/93	27/04/93	26/04/93	DELIVER IN PERSON	TRUCK	MMMMMMMMMMMMMM	1106

Figura 104 - Select na tabela BASE_LINEITEM com L_PARTKEY = 59801

É importante ressaltar que, para que a regra de autorização seja aplicada corretamente, o usuário responsável pela criação da *trigger* não pode ter o privilégio EXEMPT ACCESS POLICY. Se a *trigger* tivesse sido criada utilizando um usuário com este privilégio todas as políticas de acesso seriam ignoradas.

5.4 Oportunidades e Limitações identificadas

O label security é uma importante ferramenta para restringir acesso a registros específicos. Dado esta característica, ele funcionou corretamente para os testes das

regras T1, T3, T4 e T5, mas não pôde ser usado para tratar a regra T2, pois esta restringe o acesso a colunas específicas e não a registros inteiros, principal limitação do Label Security. Outra limitação do label security é que tem que ser criada uma coluna para cada política que se deseja registrar em uma tabela. Se existirem vários perfis com diferentes acessos, várias colunas serão criadas levando a um grande aumento de espaço de armazenamento.

6 Proposta de modelo para armazenamento das regras de autorização

A seguir é apresentada a proposta de modelo de controle de autorização (Figura 105). O modelo foi elaborado para tratar autorização de acesso ao TPC-H, mas está genérico o suficiente para ser utilizado para controle de autorização em outras bases de dados.

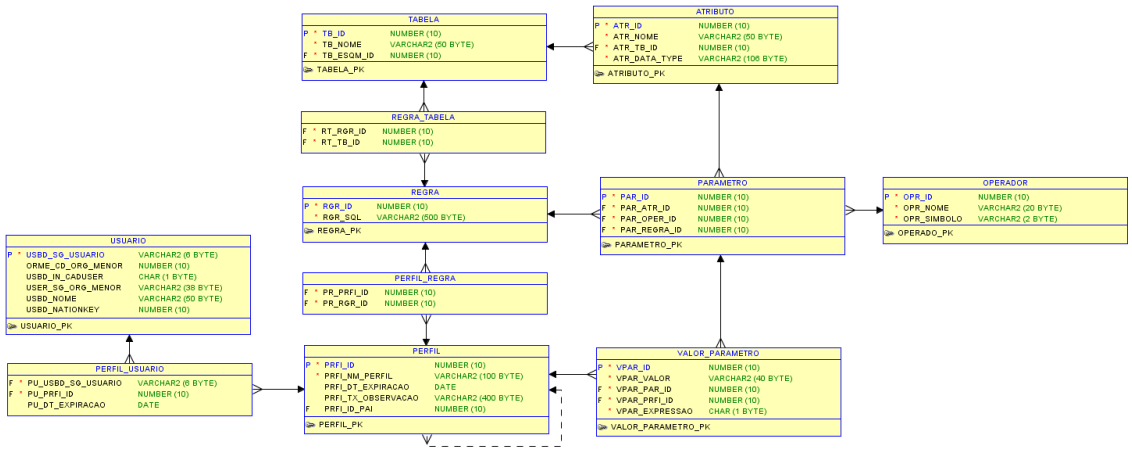


Figura 105 - Proposta de modelo de controle de autorização

As seguintes tabelas fazem parte do modelo de entidades para armazenar as regras de autorização:

- USUARIO: Essa tabela armazena os usuários que serão associados aos perfis de acesso à base de dados. Para a nossa proposta, os usuários cadastrados nessa tabela devem ser usuários do banco de dados. Isto porque, na execução da regra de autorização, recupera-se o usuário (usuário do banco de dados) que está executando a operação, por exemplo, uma consulta; e a partir deste usuário obtém-se os perfis aos quais ele está associado. A operação só é executada se um dos perfis do usuário tiver acesso para executá-la.
- PERFIL: Essa tabela armazena os perfis utilizados para aplicar as regras de acesso. No modelo proposto é possível, atráves de um auto-relacionamento, a criação de uma hierarquia entre os perfis de acesso o que permite, por exemplo, a existência de um perfil pai chamado *Gerente de vendas* e dos perfis filhos chamados *Gerente de vendas da América e Ásia* e *Gerente de vendas da Europa*. A implementação atual contempla a relação pai-filho em um único nível. No entanto, é fácil estendê-la para contemplar mais de um nível. Os níveis internos seriam utilizados para definir regras que se aplicam aos filhos. Dessa forma, na implementação atual, os perfis pais devem manter o campo do auto-relacionamento vazio, enquanto os perfis filhos devem preenchê-lo com o identificador (ID) do prefil pai. Ainda é possível criar um perfil simples, ou seja, que não é pai nem filho de nenhum outro perfil, para isso basta preencher o campo do auto relacionamento com o ID do próprio perfil.

- Em uma hierarquia de perfis, a tupla correspondente ao pai armazena à regra que deve ser aplicada aos perfis da hierarquia, já as tuplas correspondentes aos filhos definem os valores específicos dos parâmetros para o predicado da regra de acesso.

  Como exemplo, considere uma hierarquia em que o perfil pai *Gerente de vendas* tem dois perfis filhos *Gerente de vendas da América e Ásia* e *Gerente de vendas da Europa* (Figura 106). Associado ao perfil *Gerente de vendas* estará a regra que restringe o acesso aos dados de gerente de vendas , enquanto associado os perfis *Gerente de vendas da América e Ásia* e *Gerente de vendas da Europa*, estarão associados os parâmetros que expecificam mais ainda a regra, para que assim ela seja aplicada para as regiões da América e Ásia (no caso do primeiro perfil) e para a região Europa (no caso do segundo perfil).

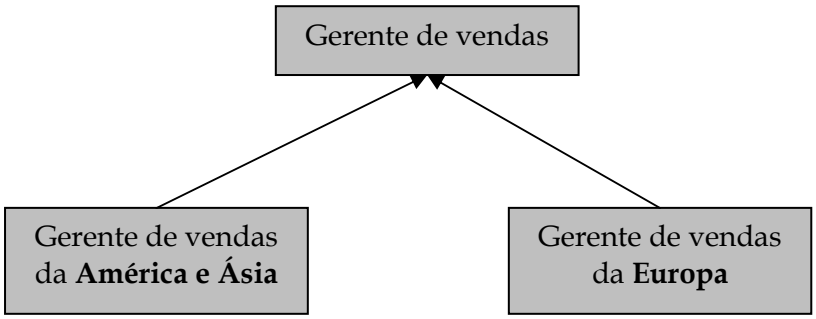


Figura 106 – Hierarquia de gerentes de vendas

- PERFIL_USUARIO: Essa tabela forma o relacionamento M × N entre as tabelas USUARIO E PERFIL, de forma que um usuário pode possuir vários perfis e cada perfil pode ser associado a vários usuários.
  - Essa tabela além de realizar o relacionamento M×N, ela também define a validade da aplicação de um perfil a um determinado usuário, através da coluna PU_DT_EXPIRACAO. O perfil do usuário é válido enquanto esta coluna estiver com valor nulo ou, caso preenchida, a data atual for menor a data que ela armazena.
  - É importante ressaltar que, como um usuário pode possuir mais de um perfil ao mesmo tempo, as regras dos diferentes perfis devem ser concatenadas. Nossa proposta considera a concatenação das regras utilizando OR, ou seja, união das regras. Por exemplo, se o usuário tem perfil que dá acesso aos pedidos dos Estados Unidos e tem um outro perfil que dá acesso aos pedidos da Europa (Figura 107), então o OR das regras permite que ele acesse os pedidos de ambas as regiões. Caso contrário, se as regras fossem concatenadas com AND, ele não teria acesso a nenhum pedido, pois a concatenação das regras com AND dá acesso apenas à interção entre perfis.

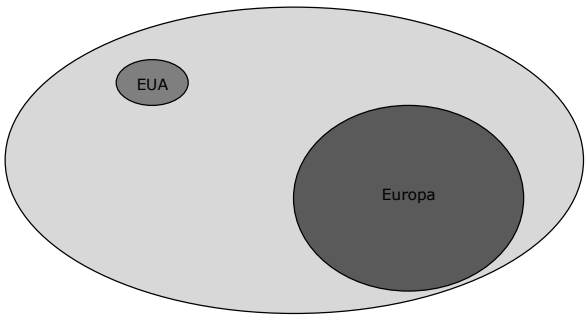


Figura 107 – Exemplo de concatenação de perfis

- REGRA: Essa tabela armazena as regras que deverão ser aplicadas a cada perfil. Nesse modelo as regras são consultas SQL que serão parte do predicado retornado pelo VPD, portanto, a regra não depende apenas do perfil do usuário que está executando a consulta, mas também da tabela sobre a qual essa consulta está sendo realizada. Como cada regra deve ser associada à dupla Perfil × Tabela, existe a possibilidade de um mesmo perfil ter diferentes regras para diferentes tabelas, analogamente existe a possibilidade de uma tabela estar associada a diversas regras de diversos perfis diferentes. A solução para esse problema é resolvido pela utilização das três próximas tabelas.
    - Em uma hierarquia de perfis, a regra associada ao perfil pai será cadastrada nessa tabela, e todos os perfis filhos vão depender dela, dessa forma, os perfis filhos não possuirão entradas nessa tabela, mas sim nas tabelas de parâmetros específicos, que serão discutidas nos itens abaixo.
- TABELA: Essa tabela armazena os nomes das tabelas que possuem políticas de acesso e ainda os nomes das tabelas que possuem algum atributo relacionado à algum parâmetro SQL específico.
- PERFIL_REGRA: Essa tabela forma o relacionamento M×N entre as tabelas PERFIL e REGRA, de forma que um perfil pode possuir várias regras de acesso, e uma mesma regra possa ser associada a vários perfis.
- REGRA_TABELA: Essa tabela forma o relacionamento M×N entre as tabelas TABELA e REGRA, de forma que uma tabela pode possuir várias regras de acesso, e uma mesma regra possa ser associada a várias tabelas.
    - O relacionamento M×N entre as tabelas TABELA e REGRA é interessante no caso em que existam várias tabelas com informações sobre as quais se deseja aplicar a regra de autorização em um mesmo perfil. Como exemplo, no modelo do TPC-H apresentado na Figura 45, as tabelas NATION, SUPPLIER e CUSTOMER têm a mesma coluna NATIONKEY que armazena a chave da nação. Caso se deseje cadastrar uma regra que restrinja o acesso a determinadas regiões para um determinado perfil (por exemplo, NATIONKEY = 1), esta regra poderia ser cadastrada para todas as três tabelas.
    - Observe que agora as tabelas PERFIL, PERFIL_REGRA, REGRA, REGRA_TABELA e TABELA permitem que um mesmo perfil tenha diferentes regras para diferentes tabelas e ainda forma a dupla Perfil × Tabela discutida anteriormente.

- PARAMETRO: Essa tabela armazena os parâmetros das regras cadastradas na tabela REGRA, esses serão os parâmetros específicos dos perfis filhos. Os parâmetro são associados à regra cadastrada na REGRA para perfil pai, indicando a qual regra o parâmetro se refere. No entanto os valores para os parâmetros são associados aos perfis filhos, indicando qual valor deve ser usado para cada perfil filho. Por exemplo, o parâmetro que define a região está associado ao perfil *Gerente de vendas*, mas os valores para os parâmetros estão associados aos perfis *Gerente de vendas da América e Ásia* e *Gerente de vendas da Europa*. Um parâmetro é formado por um atributo, um operador e o valor do parâmetro.
    - Exemplo: *n_name in ('Moçambique', 'Índia', 'Rússia')*
      Formato: *atributo operador valor*
    - Observe que para todos os perfis filhos de uma hierarquia o atributo e o operador do parâmetro serão os mesmos, a variação é apenas do valor aceito pelo parâmetro.
- ATRIBUTO: Essa tabela armazena os atributos relacionados a um parâmetro de uma regra de acesso. A tabela atributo possui uma relação com a tabela TABELA, pois um atributo é um campo presente em uma tabela.
- OPERADOR: Essa tabela que armazena os operadores existentes na SQL, são eles:
    - = (igual)
    - <> (diferente)
    - > (maior)
    - >= (maior ou igual)
    - < (menor)
    - <= (menor ou igual)
    - OR
    - IN
- VALOR_PARAMETRO: Essa tabela armazena os valores aceitos por um determinado parâmetro de uma regra de acesso. Por exemplo, os seguintes valores poderiam estar cadastrados: '*Moçambique*'; '*Índia*'; '*Rússia*'.

6.1 Função genérica para retornar os predicados para autorização

Como mencionado anteriormente, o predicado da política do VPD é definido através da execução de uma função aplicada à tabela, que retorna o predicado definido em forma de *string*. No nosso cenário, a função genérica apresentada nessa seção deve ser aplicada em cada tabela que tenha controle de autorização, através do comando apresentado na Figura 108. Observe que o parâmetro "sec_relevant_cols" é opcional e define as colunas relevantes para o controle de acesso, ou seja, o controle só será efetuado para as consultas que envolverem tais colunas. Observe também que o parâmetro "update_check", que define o comportamento do VPD em inserções e atualizações, também pode ser suprimido, nesse caso ele assumirá o valor *false*. Maiores sobre o parâmetro "update_check" pode ser encontrado nos subseções 7.2.1 e 7.2.2 desse relatório.

```
BEGIN
DBMS_RLS.ADD_POLICY ('esquema da tabela',
                     'nome da tabela',
                     'nome da política',
                     'esquema da função',
                     'nome da função',
                     sec_relevant_cols => 'colunas relevantes',
                     update_check => boolean);
END;
```

Figura 108 - Comando de aplicação da função genérica de controle de acesso

A Figura 109 apresenta a função genérica para retornar o predicado para autorização.

```
create or replace
FUNCTION "F_POLICY" (p_schema in varchar2, p_tab in varchar2) return varchar2
as
  -- Armazena o usuário do sistema que está executando a consulta
v_user varchar2 (100);

  -- Armazena o predicado dos perfis do usuário
v_return_string varchar2 (10000) default '';

  -- Armazena o número de perfis ativos na tabela
v_numero_perfis_ativos number default 0;

  -- Armazena a regra do perfil do usuário (parte do predicado sem parâmetros
  -- específicos que é associada aos perfis pais e aos perfis sem hierarquia)
v_regra regra.rgr_sql%type;

  -- Armazena a ID associada à regra do perfil do usuário
v_regra_id regra.rgr_id%type;

  -- Variável que armazena a ID associada ao perfil do usuário
v_perfil_id perfil.prfi_id%type;

  -- Recupera as regras dos perfis do usuário, as IDs associadas a essas regras e
  -- as IDs associadas a esses perfis
cursor cur_regras_perfil
is
  select rgr_sql, rgr_id, filho.prfi_id
  from regra
    -- Junta as tabelas (para formar a dupla perfil x tabela)
  inner join regra_tabela on rt_rgr_id = rgr_id
  inner join tabela on tb_id = rt_tb_id
    -- Junta os perfis pais
  inner join perfil_regra on pr_rgr_id = rgr_id
  inner join perfil pai on pai.prfi_id = pr_prfi_id
    -- Junta os perfis filhos
  inner join perfil filho on filho.prfi_id_pai = pai.prfi_id
    -- Junta o usuário
  inner join perfil_usuario on pu_prfi_id = filho.prfi_id
  inner join usuario on usbd_sg_usuario = pu_usbd_sg_usuario
  where lower (usbd_sg_usuario) = lower(v_user)
      -- Onde o usuário deve ser o usuário que faz a consulta
    and lower (tb_nome) = lower(p_tab)
      -- Onde a tabela deve ser a tabela onde a consulta é feita
    and (filho.prfi_dt_expiracao > sysdate or filho.prfi_dt_expiracao is null)
      -- Onde a data de expiração do perfil é maior que o dia de hoje ou nula
    and (pu_dt_expiracao > sysdate or pu_dt_expiracao is null)
      -- Onde a data de expiração da aplicação do perfil ao usuário é maior que o
      -- dia de hoje ou nula
    and (pai.prfi_dt_expiracao > sysdate or pai.prfi_dt_expiracao is null);
      -- Onde a data de expiração do perfil é maior que o dia de hoje ou nula


  -- Armazena os parâmetros do perfil do usuário (parte do predicado com parâmetros
  -- específicos que é associada aos perfis filhos)
v_parametros varchar2 (10000) default '';

  -- Armazena o nome da tabela que possui um campo que é atributo de um parâmetro do
  -- perfil do usuário
v_par_tabela_nome tabela.tb_nome%type;

  -- Armazena o atributo que faz parte de um parâmetro do perfil do usuário
v_par_atributo_nome atributo.atr_nome%type;

  -- Armazena o tipo do atributo que faz parte de um parâmetro do perfil do usuário
```

```
29 v_par_atributo_tipo atributo.atr_data_type%type;

   -- Armazena o símbolo de um operador que faz parte de um parâmetro do perfil
   -- do usuário
30 v_par_operador_simbolo operador.opr_simbolo%type;

   -- Recupera os atributos com seus tipos e suas tabelas de origem e os operadores
   -- que formam os parâmetros do perfil do usuário
31 cursor cur_parametros_perfil
32 is
33 select tb_nome, atr_nome, atr_data_type, opr_simbolo
34   from regra
35   inner join parametro on par_rgr_id = rgr_id
36   inner join atributo on atr_id = par_atr_id
37   inner join operador on opr_id = par_oper_id
38   inner join tabela on tb_id = atr_tb_id
39   where rgr_id = v_regra_id ;


   -- Armazena todos os valores aceitos por um parâmetro do perfil do usuário
40 v_valores_parametro varchar2 (10000) default '';

   -- Armazena um dos valores aceitos por um parâmetro do perfil do usuário
41 v_valor_parametro valor_parametro.vpar_valor%type;

   -- Armazena o indicador se o valor do parâmetro é uma expressão
42 v_valor_expressao valor_parametro.vpar_expressao%type;

   -- Recupera os valores aceitos pelos parâmetros do perfil do usuário, e o
   -- indicador do valor como expressão
43 cursor cur_valores_parametro
44 is
45   select vpar_valor, vpar_expressao
46   from valor_parametro
47   inner join parametro on par_id = vpar_par_id
48   inner join atributo on atr_id = par_atr_id
49   inner join tabela on tb_id = atr_tb_id
50   where tb_nome = v_par_tabela_nome
51     and vpar_prfi_id = v_perfil_id
52     and atr_nome =  v_par_atributo_nome;

53 begin
     -- Recupera usuário que está realizando a consulta
54   v_user := lower(sys_context('userenv', 'session_user'));
55   if v_user != 'secuser' then
56     null;
57   else
58     v_user := lower(sys_context('userenv','proxy_user'));
59   end if;

     -- Abre o predicado
60   v_return_string := '1=2 OR (';

     -- Abre o cursor de regras e recupera as regras de todos os perfis do usuário
61   open cur_regras_perfil;
62   loop
63     fetch cur_regras_perfil into v_regra, v_regra_id, v_perfil_id ;
64     exit when cur_regras_perfil%NOTFOUND;

       -- Abre o cursor de parâmetros e recupera todos os parâmetros do perfil do
       -- usuário
       -- Está em loop aninhado, então faz isso para todos os perfis do usuário
65     open cur_parametros_perfil;
       -- Limpa os parâmetros concatenados na última abertura do cursor
66     v_parametros := '';
67     loop
68       fetch cur_parametros_perfil into v_par_tabela_nome, v_par_atributo_nome,
69             v_par_atributo_tipo, v_par_operador_simbolo;
70       exit when cur_parametros_perfil%NOTFOUND;

     -- Concatena os parâmetros com 'AND' entre si
71       if length(v_parametros) is not null then
72         v_parametros := v_parametros || ' AND ';
73       end if;

         -- Abre o cursor de valores do parâmetro e recupera todos os valores aceitos
         -- pelo parâmetro do perfil do usuário
         -- Está em loop aninhado, então faz isso para todos os parâmetros do perfil
         -- do usuário
74       open cur_valores_parametro;
         -- Limpa os valores dos parâmetros concatenados na última abertura do cursor
75       v_valores_parametro := '(';
76       loop
```

```
 77         fetch cur_valores_parametro into v_valor_parametro, v_valor_expressao;
 78         exit when cur_valores_parametro%NOTFOUND;

            -- Concatena os valores dos parâmetros com ',' entre si
 79         if v_valores_parametro  <> '(' then
 80           v_valores_parametro := v_valores_parametro  || ' , ';
 81         end if;

            -- Se o valor do parâmetro não for uma expressão ou do tipo NUMBER,
            -- o valor deve ser concatenado entre '
 82         if lower(v_valor_expressao) = 's' or
 83            lower(v_par_atributo_tipo) = 'number' then
 84           v_valores_parametro := v_valores_parametro  || v_valor_parametro;
 85         else
 86           v_valores_parametro := v_valores_parametro  || '''' ||
 87                                  v_valor_parametro || '''';
 88         end if;
 89       end loop;
 90       close cur_valores_parametro;

          -- Fecha os valores do parâmetro
 91       v_valores_parametro := v_valores_parametro || ')';

          -- Monta os parâmetros do perfil
 92       v_parametros := v_parametros || v_par_atributo_nome || ' ' ||
 93                       v_par_operador_simbolo || ' ' || v_valores_parametro;

 94     end loop;
 95     close cur_parametros_perfil;

        -- Concatena os predicados dos perfis com 'OR' entre si
 96     if v_return_string <> '1=2 OR (' then
 97       v_return_string := v_return_string || ' OR ';
 98     end if;

        -- Monta o predicado do perfil
        -- Se existir uma regra para o pai, o parênteses deve ser fechado no fim do
        -- predicado
 99     if length(v_regra) is not null then
100       v_return_string :=  v_return_string || v_regra || v_parametros || ')';
101     else
102       v_return_string :=  v_return_string || v_parametros;
103     end if;

104   end loop;
105   close cur_regras_perfil;

      -- Fecha o predicado
106   v_return_string := v_return_string || ')';

      -- Substitui o coringa ?user pelo usuário que está executando a consulta.
107   v_return_string := replace(v_return_string, '?user', v_user);

      -- Verifica se o usuário não possui nenhum perfil
108   if v_return_string = '1=2 OR ()'  then
        -- Recupera o número de perfis ativos associados à tabela onde o comando foi
        -- executado
109     select count(*) into v_numero_perfis_ativos
110     from perfil
111     inner join perfil_regra on pr_prfi_id = prfi_id
112     inner join regra on rgr_id = pr_rgr_id
113     inner join regra_tabela on rt_rgr_id = rgr_id
114     inner join tabela on tb_id = rt_tb_id
115     where lower(tb_nome) = lower(p_tab)
116       and (prfi_dt_expiracao > sysdate or prfi_dt_expiracao is null);

        -- Se a tabela possuir algum perfil ativo associado à ela, retorna '1=2' senão
        -- retorna  '1=1'
117     if (v_numero_perfis_ativos > 0) then
118       v_return_string := '1=2';
119     else
120       v_return_string := '1=1';
121     end if;

122   end if;

123   return v_return_string;
124 end;
```

Figura 109 – Função de aplicação da regra de autorização

O primeiro passo executado pela função genérica é a captura do usuário que está executando a consulta. Em seguida, para cada perfil do usuário, é montado o predicado, unindo as informações cadastradas nas tabelas do modelo de armazenamento das regras de autorização. Caso o perfil do usuário não possua um predicado específico para aquela tabela, deve-se checar se a tabela tem alguma outra restrição cadastrada válida, o que significa que a tabela tem acesso controlado. Caso a tabela tenha um predicado válido então é retornado "1=2", o que bloqueia o acesso a todo o conteúdo da tabela. Não possuindo nenhum predicado, então é retornado "1=1", liberando o acesso à tabela.

No primeiro bloco da função, compreendido entre as linhas 4 e 52, encontram-se as declarações das variáveis e dos cursores. Cada variável e cursor possui um comentário com uma breve descrição, contudo é interessante detalharmos um pouco mais os cursores.

O primeiro cursor chamado de *cur_regras_perfil* recupera as regras e os perfis do usuário (linhas 10 a 25). Antes de analisarmos esse cursor, é importante lembrar que as regras do usuário dependem da dupla Perfil × Tabela, portanto a consulta deve incluir essas duas tabelas, ou seja, na execução da função devem ser considerados os perfis do usuário que estão associados à tabela a qual a regra será executada. A consulta do cursor parte da tabela REGRA (linha 13) e então é feita uma junção com a tabela REGRA_TABELA (linha 14) e outra junção com a tabela TABELA (linha 15), dessa forma obtendo os perfis que estão associados à tabela considerada para aplicação das regras (explicitada no predicado *lower (tb_nome) = lower(p_tab)* – linha 22). Com a junção com as tabelas PERFIL_REGRA (linha 16) e mais uma junção com a tabela PERFIL (linha 17), a partir desse momento já é possível fazer a relação Perfil × Tabela. Contudo, como dito na sessão **Erro! Fonte de referência não encontrada.**, a tabela REGRA recebe uma entrada para o perfil pai da hierarquia, enquanto o usuário está associado a um perfil filho, desse modo, é necessário montar a hierarquia para podermos achar o perfil pai do perfil do usuário. Para isso é feita mais uma junção com a tabela PERFIL (linha 18), formando a hierarquia, e uma junção com a tabela USUARIO, para que possamos capturar apenas o perfil do usuário que está realizando a consulta (linhas 19 e 20). A última parte da consulta são os parâmetros, o primeiro especifica o usuário que está realizando a consulta (linha 21), o segundo especifica a tabela onde a consulta é realizada (linha 22) (formando a dupla Perfil × Tabela), o terceiro verifica se o perfil ainda está válido (linha 23), o quarto verifica se a aplicação do perfil ao usuário ainda está válida (linha 24) e finalmente, o quinto verifica se o perfil pai ainda está válido (linha 25).

O segundo cursor, chamado *cur_parametros_perfil*, captura os componentes dos parâmetros do perfil pai do perfil do usuário (linha 31). Lembrando que os parâmetros do perfil são associados à regra cadastrada da tabela REGRA para o perfil pai; portanto, essa tabela deve estar incluída na consulta. A consulta do cursor parte da tabela REGRA (linha 34) e então faz uma junção com a tabela PARAMETRO (linha 35), capturando todos os parâmetros para aquela REGRA. Como o parâmetro é composto por atributo, operador e valores, devem ser feitas mais três junções, uma com a tabela ATRIBUTO (linha 36), outra com a tabela OPERADOR (linha 37) e mais uma com a tabela TABELA (linha 38). Essa última junção é necessária para obtermos o nome da tabela do atributo do parâmetro, que será usada mais tarde para recuperar dinamicamente os valores aceitos pelo parâmetro. Finalmente a consulta é encerrada com um parâmetro que especifica de qual REGRA são os parâmetros que desejamos recuperar (linha 39).

O terceiro cursor, chamado *cur_valores_parametro*, captura os valores dos parâmetros do perfil do usuário(linha 43). Estes valores são utilizados para completar o predicado

que deve ser aplicado para a autorização. A consulta do cursos parte da tabela VALOR_PARAMETRO (linha 46). Em seguida, faz junção com a tabela PARAMETRO (linha 47) a fim de recuperar os valores do parâmetro correto, o que é indicado pela variável vpar_par_id. Em seguida, descobre-se de qual atributo e de qual tabela está se buscando o valor, através das junções com as tabelas ATRIBUTO e TABELA (linhas 48 e 49). Para tal, é necessário indicar qual é a tabela (linha 50), qual é o nome do atributo (linha 52) e de qual perfil está-se buscando o valor (linha 51).

O segundo bloco da função, compreendido entre as linhas 53 e 124, monta o predicado da política propriamente dito. O primeiro passo é capturar o usuário que realizou a consulta, isso é feito utilizando a função *sys_context* (linhas 54 a 59). Em seguida o predicado é iniciado com "1=2 OR (" (linha 60), o cursor *cur_regras_perfil* é aberto (linha 61), e entramos em um loop onde será montado o predicado (linha 62). O loop percorrerá todos as regras de perfil cadastrados para esse usuário na tabela REGRA, mas só haverá mais de um se o usuário tiver mais de um perfil com regra de acesso para a tabela sobre a qual a restrição está sendo considerada. Dentro do loop é aberto o cursor *cur_parametros_perfil* (linha 65). Em seguida, entramos em outro loop para recuperar todos os parâmetros do perfil (linha 67). O loop é aninhado porque, caso o usuário tenha mais de uma regra de perfil, os parâmetros terão que ser montados para cada um deles, portanto, antes de entrar no segundo loop, é necessário também limpar a string que armazena os parâmetros montados (linha 66).

Dentro do segundo loop a string com os parâmetros é montada concatenando o atributo, o operador e os valores aceitos pelo parâmetro. Os valores para os parâmetros são obtidos através do cursor *cur_valores_parametro* (linha 74). Um loop é executado neste cursor (linha 76), concatenando cada valor de acordo se o valor corresponde a uma expressão (linha 82), número (linha 83) ou qualquer outro tipo (por exemplo, string). Além disso, os parâmetros do perfil são concatenados por AND (linha 72). Ao sair do segundo loop, o predicado é montado concatenando a regra do perfil com os parâmetros do perfil (linha 100) no caso de existir uma regra de perfil, ou concatenando somente os parâmetros do perfil (linha 102) no caso de não existir uma regra de perfil. Observe que, caso o usuário possua mais de um perfil para a tabela, as regras dos perfis serão concatenadas por OR (linhas 96 a 98).

Caso o id do usuário está sendo utilizado na regra, através da utilização do coringa "?user", então é feita a substituição deste valor pelo id do usuário (linha 107) recuperado do contexto de execução da função.

Finalmente, se o usuário não possuir nenhum Perfil com regra de acesso para essa tabela (linha 108), será feita uma consulta para verificar se existe algum perfil ativo que possua regra de acesso para a tabela (linhas 109 a 116). Se tal perfil existir, será retornado o predicado "1=2", impedindo o acesso a todas as linhas da tabela, caso contrário será retornado o predicado "1=1", liberando o acesso a todas as linhas.

6.2 Função para mascaramento de colunas

A função apresentada no item anterior é utilizada quando o controle de acesso deve restringir o acesso a algumas entradas da tabela, contudo, como foi apresentado na sessão de uso da funcionalidade (2.1.2), através do VPD é possível ainda limitar o acesso aos dados por coluna através de uma função de mascaramento, porém tanto a aplicação da função quanto a própria função diferem do caso anterior. No nosso cenário, a função de mascaramento apresentada nessa seção deve ser aplicada em cada coluna de cada tabela que tenha controle de autorização, através do comando apresentado na Figura 110.

```
BEGIN
DBMS_RLS.ADD_POLICY ('esquema da tabela',
                     'nome da tabela',
                     'nome da política',
                     'esquema da função'
                     'nome da função',
                     sec_relevant_cols=>'colunas relevantes',
                     sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS);
END;
```

Figura 110 - Comando de aplicação da função de mascaramento de colunas

6.3 Extensão do modelo para gerar função de criação/remoção de política

A atribuição da política a uma tabela é feita através do comando "DBM_RLS.ADD_POLICY", como exemplificado na Figura 110, enquanto que a remoção é feita utilizando o comando "DBM_RLS.DROP_POLICY", montado de forma semelhante ao comando de criação de política. A princípio, estes comandos deveriam ser digitados pelo desenvolvedor responsável pelo cadastro e remoção da política. No entanto, propomos evoluir o modelo apresentado na Figura 109, incluindo novas tabelas para armazenar as informações necessárias para montar o comando dinamicamente. Este modelo é apresentado na Figura 111. Em seguida, as tabelas correspondentes a esta extensão são apresentadas em detalhes e exemplos de cadastro de informações nestas tabelas e geração dos comandos de criação e remoção são apresentados.

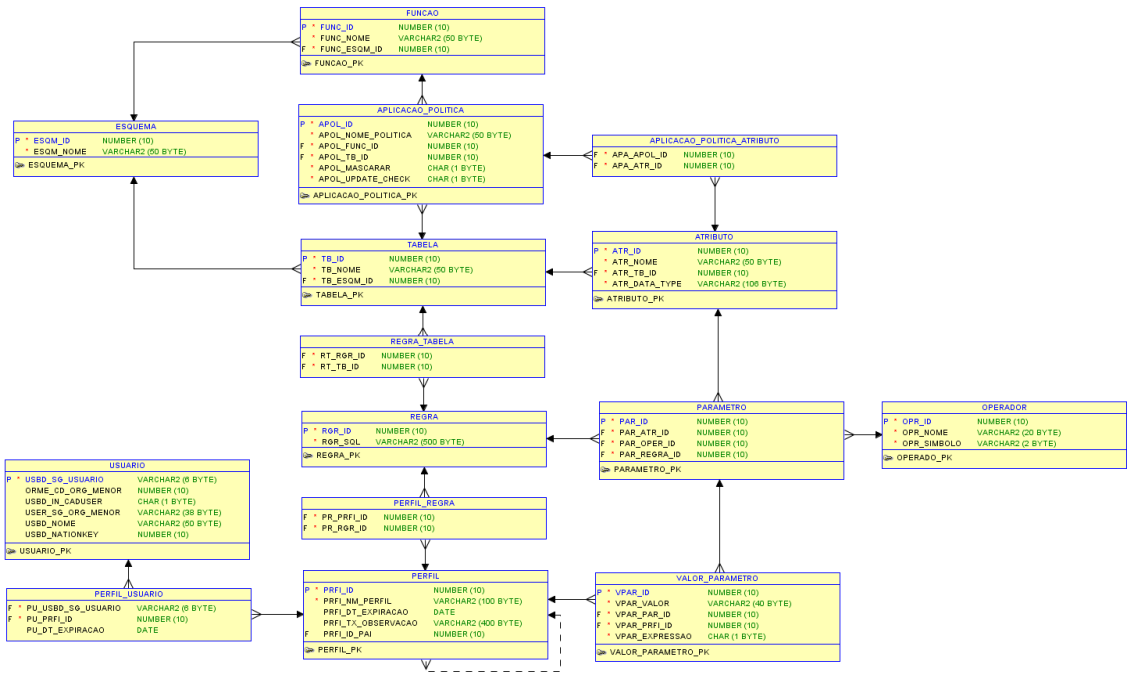


Figura 111 – Extensão do modelo proposto para incluir informações para montagem dos comentados de criação e remoção de política

As seguintes tabelas foram incluídas no modelo para armazenar as informações necessárias para montar os comandos de criação e remoção das políticas:

- ESQUEMA: Essa tabela armazena os esquemas das tabelas que possuem controle de acesso, além dos esquemas onde as funções de autorização de acesso foram definidas.

- FUNCAO: Essa tabela armazena as funções de autorização de acesso criadas. Ela poossui um relacionamento com a tabela ESQUEMA que indica o esquema no qual a função foi definida.

- TABELA: Essa tabela já existia no modelo de armazenamento das regras de autorização, mas recebeu um novo campo chamado TB_ESQM_ID. Esse campo define um relacionamento com a tabela ESQUEMA, indicado a qual esquema a tabela pertence.

- APLICACAO_POLITICA: Essa tabela armazena os nomes das políticas aplicadas às tabelas. Cada tabela pode possuir várias políticas e cada uma dessas políticas pode ser associada a uma função diferente. Para representar isso a tabela APLICACAO_POLITICA define um relacionamento M×N entre as tabelas TABELA e FUNCAO.

    - Como mencionado anteriormente, uma política possui a propriedade opcional de mascarar as colunas relevantes. Para definir a utilização ou não dessa propriedade pelo VPD, foi criado o campo APOL_MASCARAR, que deve receber o valor 'S' quando a propriedade for utilizada pela política e o valor 'N' quando não for utilizada.

    - Outra propriedade que uma política pode utilizar é o UPDATE_CHECK. Essa propriedade recebeu a mesma abordagem da anterior. Nesse caso foi criado o campo APOL_UPDATE_CHECK, que deve receber o valor 'S' quando a propriedade for utilizada pela política e o valor 'N' quando não for utilizada. Essa propriedade é melhor descrita nas sessões 7.2.1 e 7.2.2 .

- APLICACAO_POLITICA_ATRIBUTO: Essa tabela cria o relacionamento M×N entre a política e os atributos da tabela, definindo as colunas relevantes para o controle de acesso.

A partir das informações cadastradas neste modelo, utilizando a função apresentada na Figura 112, gera-se o comando para criação da política (com nome nome_política, passando tipo com valor ‘c’) e para remoção da política (com nome nome_política, passando tipo com valor ‘r’).

```
create or replace
FUNCTION F_COMANDO_POLITICA (nome_politica in varchar2, tipo_comando in char)
return varchar2 as
-- Armazena o comando de criação/remoção da política que será retornado
-- pela função
v_comando varchar2 (1000) default'';

-- Armazena o nome do esquema da tabela
v_tabela_esquema varchar2 (50);

-- Armazena o nome da tabela
v_tabela_nome varchar2 (50);

-- Armazena o nome do esquema da função
v_funcao_esquema varchar2 (50);

-- Armazena o nome da função
```

```
v_funcao_nome varchar2 (50);

-- Armazena o indicador da opção de mascaramento da política
v_mascarar varchar (50);

-- Armazena o indicador da opção de update check da política
v_update_check varchar2 (5);

-- Armazena o nome de uma coluna relevante da política
v_coluna_relevante varchar (50);

-- Armazena o nome de todas as colunas relevantes da política
v_colunas_relevantes varchar2 (500) default'';

-- Captura as colunas relevantes da política
cursor cur_colunas_relevantes
is
  select atr_nome
  from atributo
  inner join aplicacao_politica_atributo on atr_id = apa_atr_id
  inner join aplicacao_politica on apa_apol_id = apol_id
  where lower(apol_nome_politica) = lower(nome_politica);

begin
  -- Captura os valores dos parâmetros do comando de criação/remoção
  -- da política
  select apol_mascarar, apol_update_check, q1.esqm_nome, func_nome,
         q2.esqm_nome, tb_nome into v_mascarar, v_update_check,
         v_funcao_esquema, v_funcao_nome, v_tabela_esquema, v_tabela_nome
  from aplicacao_politica
  inner join funcao on func_id = apol_func_id
  inner join tabela on tb_id = apol_tb_id
  inner join esquema q1 on func_esqm_id = q1.esqm_id
  inner join esquema q2 on tb_esqm_id = q2.esqm_id
  where lower(apol_nome_politica) = lower(nome_politica);

  -- Se a opção de update check for marcada como "Sim",
  -- define o valor do parâmetro como "true"
  -- Se a opção de update check for marcada como "Não",
  -- define o valor do parâmetro como "false"
  if v_update_check = 'S' then
    v_update_check := 'true';
  else
    v_update_check := 'false';
  end if;

  -- Se a opção de mascarar colunas relevantes for marcada como "Sim",
  -- define o parâmetro extra "sec_relevant_cols_opt => dbms_rls.ALL_ROWS"
  -- Se a opção de mascarar colunas relevantes for marcada como "Não",
  -- não define nenhum parâmetro extra
  if v_mascarar = 'S' then
    v_mascarar := ', sec_relevant_cols_opt => dbms_rls.ALL_ROWS';
  else
    v_mascarar := '';
  end if;

  -- Monta a string com os nomes das colunas relevantes
  open cur_colunas_relevantes;
  loop
    fetch cur_colunas_relevantes into v_coluna_relevante;
    exit when cur_colunas_relevantes%NOTFOUND;

    if length(v_colunas_relevantes) is not null then
        v_colunas_relevantes := v_colunas_relevantes || ', ';
    end if;

    v_colunas_relevantes := v_colunas_relevantes || v_coluna_relevante;

  end loop;
  close cur_colunas_relevantes;

  -- Se a política possuir alguma coluna relevantes, especificar o
  -- parâmetro "sec_relevant_cols =>" antes dos nomes das colunas
```

```
  if length(v_colunas_relevantes) is not null then
       v_colunas_relevantes := ', sec_relevant_cols => ''' ||
                               v_colunas_relevantes || '''';
  end if;

  -- Monta o comando.
   -- 'c' - Criação
   -- 'r' - Remoção
  if lower(tipo_comando) = 'c' then
    v_comando := 'dbms_rls.add_policy(' ||
       'object_schema => ''' || v_tabela_esquema ||
       ''', object_name => ''' || v_tabela_nome ||
       ''', policy_name => ''' || nome_politica ||
       ''', function_schema => ''' || v_funcao_esquema ||
       ''', policy_function => ''' || v_funcao_nome ||
       ''', update_check => ' || v_update_check ||
      v_colunas_relevantes || v_mascarar || ');';
    else if lower(tipo_comando) = 'r' then
      v_comando := 'dbms_rls.drop_policy(' ||
         'object_schema => ''' || v_tabela_esquema ||
         ''', object_name => ''' || v_tabela_nome ||
         ''', policy_name => ''' || nome_politica || ''');';
    end if;
  end if;
  return v_comando;

end;
```

Figura 112 – Função para montar comandos de criação e remoção de política

A seguir são apresentados dois exemplos de informações cadastradas nas tabelas e exemplos de chamadas da função para retornar o comando de criação e o comando de remoção.

O primeiro exemplo é para a regra T1. Nesse exemplo foi feito um registro na tabela APLICACAO_POLITICA com o nome da política definido como "ACCESS_POLICY_T1". A função utilizada pela política e cadastrada na tabela FUNCAO chama-se "F_POLICY", e foi associada ao esquema "PERFIL", cadastrado na tabela ESQUEMA. A tabela onde a política se aplica e cadastrada na tabela TABELA chama-se "LINEITEM", e foi associada ao esquema "TPCH", cadastrado na tabela ESQUEMA.

A propriedade de mascaramento foi definida como 'N' enquanto a propriedade UPDATE_CHECK foi definida como 'S'. Nenhuma coluna relevante foi associada à política.

A chamada da função para retornar o comando de criação, e o comando de criação retornado para a política "ACCESS_POLICY_T1" são apresentado na Figura 113. A chamada da função para retornar o comando de remoção, e o comando de remoção retornado para a política "ACCESS_POLICY_T1" são apresentado na Figura 114.

```
select f_comando_politica ('ACCESS_POLICY_T1', 'C') from dual;

dbms_rls.add_policy(
  object_schema => 'TPCH',
  object_name => 'LINEITEM',
  policy_name => 'ACCESS_POLICY_T1',
  function_schema => 'PERFIL',
  policy_function => 'F_POLICY',
  update_check => true);
```

Figura 113 – Chamada da função de retorno do comando de criação e comando de criação para a política "ACCESS_POLICY_T1"

```
select f_comando_politica ('ACCESS_POLICY_T1', 'R') from dual;
dbms_rls.drop_policy(
  object_schema => 'TPCH',
  object_name => 'LINEITEM',
  policy_name => 'ACCESS_POLICY_T1');
```

Figura 114 – Chamada da função de retorno do comando de remoção e comando de remoção para a política "ACCESS_POLICY_T1"

O segundo exemplo de cadastro de informações para geração dos comandos foi realizado para a regra T2. Para esse segundo caso foi feito um registro na tabela APLICACAO_POLITICA com o nome da política definido como "ACCESS_POLICY_T2". A função utilizada pela política e cadastrada na tabela FUNCAO chama-se "F_POLICY_COLUMN", e foi associada ao esquema "PERFIL", cadastrado na tabela ESQUEMA. A tabela para essa política é a mesma do exemplo anterior (LINEITEM).

A propriedade UPDATE_CHECK foi definida como 'S', assim como a propriedade de mascaramento. Por fim os atributos "SHIPDATE", "SHIPINSTRUCT" e "SHIPMODE" cadastrados na tabela ATRIBUTO foram definidos como colunas relevantes, através do relacionamento criado na tabela APLICACAO_POLITICA_ATRIBUTO.

A chamada da função para retornar o comando de criação, e o comando de criação retornado para a política "ACCESS_POLICY_T2" são apresentado na Figura 115. A chamada da função para retornar o comando de remoção, e o comando de remoção retornado para a política "ACCESS_POLICY_T2" são apresentado na Figura 116.

```
select f_comando_politica ('ACCESS_POLICY_T2', 'C') from dual;
dbms_rls.add_policy(
  object_schema => 'TPCH',
  object_name => 'LINEITEM',
  policy_name => 'ACCESS_POLICY_T2',
  function_schema => 'PERFIL',
  policy_function => 'F_POLICY_COLUMN',
  update_check => true,
  sec_relevant_cols => 'SHIPDATE, SHIPINSTRUCT, SHIPMODE',
  sec_relevant_cols_opt => dbms_rls.ALL_ROWS);
```

Figura 115 – Chamada da função de retorno do comando de criação e comando de criação para a política "ACCESS_POLICY_T2"

```
select f_comando_politica ('ACCESS_POLICY_T2', 'R') from dual;
dbms_rls.drop_policy(
  object_schema => 'TPCH',
  object_name => 'LINEITEM',
  policy_name => 'ACCESS_POLICY_T2');
```

Figura 116 – Chamada da função de retorno do comando de remoção e comando de remoção para a política "ACCESS_POLICY_T2"

7 Avaliação experimental do modelo proposto em relação ao VPD

Esta seção descreve a avaliação experimental do modelo proposto na seção 5 para os exemplos de regras de autorização construídos para o Benchmark TPC-H, apresentados na seção 4

7.1 Informações cadastradas na base de dados de autorização de acesso

As seguintes informações foram cadastradas na base de dados de autorização de acesso para considerar cada teste realizado:

- Para a regra T1, cuja proposta é restringir o acesso aos itens de pedidos provenientes de fornecedores das nações do hemisfério Norte e das regiões Ásia e América, foi criada uma hierarquia de perfis onde o perfil pai chama-se *Gerente de vendas*. Além disso, para esse exemplo, também foi criado um perfil filho chamado *Gerente de vendas Ásia e América*.

  Para o perfil pai desse exemplo, foi cadastrada uma consulta SQL na tabela REGRA que, por sua vez, foi associada à tabela LINEITEM (através da tabela REGRA_TABELA), formando assim a dupla Perfil × Tabela. A consulta SQL cadastrada para o perfil pai é representada na Figura 117.

```
l_suppkey in (
  select s_suppkey
  from supplier, nation, region
  where s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and
```

Figura 117 - Consulta SQL de restrição de acesso do perfil Gerente de vendas para tabela LINEITEM

  Para o perfil filho desse exemplo foi necessária a criação de dois parâmetros:

  - Parâmetro referente ao hemisfério

    O primeiro parâmetro do perfil filho, diz respeito à restrição para o hemisfério norte. Para montar esse parâmetro, foi feito um cadastro na tabela ATRIBUTO com o valor *n_hemisphere* que, por sua vez, foi associado à tabela NATION. Em seguida foi incluído um registro na tabela PARAMETRO que efetivamente monta o parâmetro, ao referenciar o atributo criado, o operador IN (pré-cadastrado na tabela OPERADOR) e ainda a consulta SQL do perfil pai.

    Nesse momento já se possui o atributo e o operador do parâmetro da consulta SQL, faltando apenas definir os valores aceitos. Para isso foi feita uma inserção na tabela VALOR_PARAMETRO com o valor *1* (Norte), associado ao parâmetro recém criado e ao perfil filho.

  - Parâmetro referente às regiões

    O segundo parâmetro do perfil filho, diz respeito à restrição para as regiões Ásia e América. Para montar esse parâmetro, foi feito um cadastro na tabela ATRIBUTO com o valor *r_regionkey* que, por sua vez,

foi associado à tabela REGION. Em seguida foi incluído um registro na tabela PARAMETRO que efetivamente monta o parâmetro, ao referenciar o atributo criado, o operador IN (pré-cadastrado na tabela OPERADOR) e ainda a consulta SQL do perfil pai.

Para a definição dos valores aceitos, foram feitos duas inserções na tabela VALOR_PARAMETRO, um com valor *1* (América) e outro com o valor *2* (Ásia), além de é claro, associá-los ao parâmetro recém criado e ao perfil filho.

- Para a regra T2, cuja proposta é restringir o acesso às informações de embarque contidos na tabela item de pedido, foi utilizada a mesma hierarquia de perfis do exemplo T1. No entanto, esta regra restringe o acesso a informações de colunas da tabela LINEITEM e não a registros. Este é um exemplo de mascaramento de valores de colunas, ou seja, o usuário tem acesso a determinadas informações de todos os registros, mas tem restrições para acessar outras informações. A proposta de modelo apresentada na seção 6 não trata este caso, sendo necessário criar uma função específica para ele. Após a criação da função, ela foi atribuída à tabela LINEITEM definindo quais colunas deveriam ser mascaradas quando acessadas pelo o usuário. A associação da política à tabela é apresentada na Figura 118 e a função responsável por testar a política de acesso é apresentada na Figura 119.

```
begin
dbms_rls.add_policy (
  object_schema => 'TPCH',
  object_name => 'BASE_LINEITEM',
  policy_name => 'F_POLICY_COLUMN',
  function_schema => 'PERFIL',
  policy_function => 'F_COLUMN',
  sec_relevant_cols => 'SHIPDATE, SHIPINSTRUCT, SHIPMODE',
  sec_relevant_cols_opt  => DBMS_RLS.ALL_ROWS);
end;
```

Figura 118 – Associação de política de acesso com mascaramento de colunas

```
create or replace
FUNCTION F_POLICY_COLUMN (schema in varchar2, p_tab in varchar2)
return varchar2 as

-- Armazena o usuário do sistema que está executando a consulta
v_user varchar2 (100);

-- Indica se o usuário possui o perfil especificado
v_possui_perfil varchar2 (1);

-- Armazena o predicado dos perfis do usuário
v_return_string varchar2(5) default '';

begin
  v_user := lower (sys_context ('userenv', 'session_user'));
  if v_user != 'secuser' then
      null;
  else
      v_user := lower(sys_context('userenv','proxy_user'));
  end if;

  begin
```

```
    select
     case
       when exists(
         select 1
         from perfil_usuario pu
         inner join perfil pa on pu.pu_prfi_id = pa.prfi_id
         inner join perfil pai on pai.prfi_id = pa.prfi_prfi_id_pai
         where pai.prfi_nm_perfil= 'gerente de vendas'
           and lower(pu.pu_usbd_sg_usuario) = v_user) then 't'
       else 'f'
       end
    into v_possui_perfil
    from dual;
  exception
    when no_data_found then
      v_possui_perfil := 'F';
  end;

  if v_possui_perfil = 'T' then
     v_return_string := '1=2';
  else
     v_return_string :='1=1';
  end if;

  return v_return_string;
end;
```

Figura 119 – Função específica para regra T2

- Para a regra T3, cuja proposta é restringir o acesso aos itens de pedido para clientes cadastros na aplicação APL1 e das nações Moçambique, Índia ou Rússia, foi criada uma hierarquia de perfis onde o perfil pai chame-se *Cliente*. Além disso, para esse exemplo, também foi criado um perfil filho chamado *Cliente Moçambique Índia Rússia*.

  Para o perfil pai desse exemplo, foi cadastrada uma consulta SQL na tabela REGRA que, por sua vez, foi associada à tabela LINEITEM (através da tabela REGRA_TABELA), formando assim a dupla Perfil × Tabela. A consulta SQL cadastrada para o perfil pai está representada na Figura 120.

```
l_orderkey in (
  select o_orderkey
  from orders, customer, nation
  where lower(c_name) = lower(?user)
     or lower(c_name) = lower(?user)
    and c_custkey = o_custkey
    and n_nationkey = c_nationkey
    and
```

Figura 120 - Consulta SQL de restrição de acesso do perfil Cliente para tabela LINEITEM

Para o perfil filho desse exemplo foi necessária a criação de apenas um parâmetro:

- Parâmetro referente à nação

  O parâmetro do perfil filho, diz respeito à restrição para as nações Moçambique, Índia e Rússia. Para montar esse parâmetro, foi feito uma inserção na tabela ATRIBUTO com o valor *n_name* que, por sua vez, foi associado à tabela NATION. Em seguida foi feita uma inserção na

tabela PARAMETRO que efetivamente monta o parâmetro, ao referenciar o atributo criado, o operador IN (pré-cadastrado na tabela OPERADOR) e ainda a consulta SQL do perfil pai.

Para a definição dos valores aceitos, foram feitos três cadastros na tabela VALOR_PARAMETRO, um com valor *mozambique,* outro com o valor *india* e mais um com o valor *russia*, além de é claro, associá-los ao parâmetro recém criado e ao perfil filho.

- Para a regra T4, cuja proposta é restringir o acesso aos dados de fornecedores para os usuários da mesma nação do fornecedor, foi criada uma hierarquia de perfis onde o perfil pai chame-se Gestor de depósito. Além disso, para esse exemplo, também foi criado um perfil filho chamado Gestor de depósito local.

  Como os dados dos fornecedores podem ser encontrados em duas tabelas distintas, o perfil pai necessitou de duas inserções de consultas SQL na tabela REGRA, uma para cada tabela. A primeira consulta, representada na Figura 121, foi associada à tabela SUPPLIER, enquanto a segunda consulta, representada na Figura 122, foi associada à tabela PARTSUPP. Observe que nesse caso foram formadas duas duplas Perfil × Tabela.

```
s_nationkey in (
  select usbd_nationkey
  from usuario
  where s_nationkey = usbd_nationkey
    and lower(usbd_sg_usuario) =
        lower (sys_context('userenv', 'session_user'))
     or lower(usbd_sg_usuario) =
        lower(sys_context('userenv','proxy_user'))
```

Figura 121 - Consulta SQL de restrição de acesso do perfil Gestor de depósito para a tabela SUPPLIER

```
ps_suppkey in (
  select s_suppkey
  from supplier
  inner join base_nation on n_nationkey = s_nationkey
  where s_nationkey in (
    select usbd_nationkey
    from usuario
    where s_nationkey = usbd_nationkey
      and lower(usbd_sg_usuario) =
          lower (sys_context ('userenv', 'session_user'))
       or lower(usbd_sg_usuario) =
          lower(sys_context('userenv','proxy_user')))
```

Figura 122 - Consulta SQL de restrição de acesso do perfil Gestor de depósito para a tabela PARTSUPP

Para esse exemplo não houve necessidade de criação de parâmetros para o perfil filho, ele foi usado apenas para aplicar a regra de acesso aos usuários.

É importante ressaltar que cada consulta de autorização será executada apenas quando a tabela para a qual foi cadastrada for acessada. Neste caso, a consulta da Figura 121 será executada quando do acesso à tabela SUPPLIER; enquanto que a consulta da Figura 122 será executada quando do acesso à tabela PARTSUPP.

- Para a regra T5, cuja proposta é permitir que usuários da nação Romênia possam acessar os itens de pedido de clientes das nações Brasil e Argentina, foi criada uma hierarquia de perfis onde o perfil pai chama-se Marketing. Além disso, para esse exemplo, também foi criado um perfil filho chamado Marketing Romênia.

  Assim como no exemplo anterior, os dados dos pedidos dos clientes podem ser encontrados em duas tabelas distintas, o perfil pai necessitou de dois cadastros de consultas SQL na tabela REGRA, uma para cada tabela. A primeira consulta, representada na Figura 123, foi associada à tabela ORDERS, enquanto que a segunda consulta, representada na Figura 124, foi associada à tabela LINEITEM. Novamente nesse caso foram formadas duas duplas Perfil × Tabela.

```
o_custkey in (
  select c_custkey
  from customer, nation
  where n_nationkey = c_nationkey
    and
```

Figura 123 - Consulta SQL de restrição de acesso do perfil Marketing para a tabela ORDERS

```
l_orderkey in (
  select o_orderkey
  from orders
```

Figura 124 - Consulta SQL de restrição de acesso do perfil Marketing para a tabela LINEITEM

Para o perfil filho desse exemplo foi necessária a criação de apenas um parâmetro, mas esse parâmetro vale apenas para a consulta SQL associada à tabela ORDERS, pois a consulta SQL associada à tabela LINEITEM não precisa de parâmetros extras.

- Parâmetro referente á nação

  O parâmetro do perfil filho, diz respeito à restrição para as nações Brasil e Argentina. Para montar esse parâmetro, foi feita uma inserção na tabela ATRIBUTO com o valor *n_name* que, por sua vez, foi associado à tabela NATION. Em seguida foi feita uma inserção na tabela PARAMETRO que efetivamente monta o parâmetro, ao referenciar o atributo criado, o operador *IN* (pré-cadastrado na tabela OPERADOR) e ainda a consulta SQL para a tabela ORDERS do perfil pai.

  Para a definição dos valores aceitos, foram feitos duas inserções na tabela VALOR_PARAMETRO, um com valor *brazil* e outro com o valor *argentina*, além de é claro, associá-los ao parâmetro recém criado e ao perfil filho.

É interessante observar que, apesar de não possuir parâmetros, a regra de acesso da tabela LINEITEM se aproveita da regra cadastrada na tabela

ORDERS. Observe que a consulta do predicado da tabela LINEITEM é uma simples projeção da tabela ORDERS, contudo, como já existe o controle de acesso na tabela ORDERS (implementado por esse mesmo perfil), a restrição se propaga para a tabela LINEITEM.

7.2 Análise do comportamento do VPD

Os exemplos apresentados anteriormente utilizam o VPD apenas em seleções, contudo é necessário realizar uma análise do comportamento do VPD para outras operações, tais como, inserção, atualização, remoção, operação *merge*, subconsultas, consultas com agrupamentos (*group by*), consultas com agrupamentos com cláusula *having* e *triggers*. Os próximos itens abordam o comportamento do VPD observado em cada caso, apresentando os testes realizados e os resultados obtidos.

Os testes também utilizaram o benchmark TPC-H e a proposta de modelo de controle de acesso. Além disso, para garantir que o VPD não influenciasse os resultados dos testes quando fosse considerado um usuário sem a aplicação do controle de acesso, foi concedido a este usuário o privilégio EXEMPT ACCESS POLICY, que permite que o usuário ignore todas as políticas de acesso.

7.2.1 Inserção

O objetivo deste teste é verificar o funcionamento do VPD para um comando de inserção. Deve-se ressaltar que no caso da inserção, o comando não possui cláusula *where*. Neste caso, controle de acesso trabalha de modo a não deixar que o usuário insira um registro que ele não possa ver.

Para este teste foi utilizada a tabela *supplier*, e considerado um usuário com o perfil *Gestor de Depósito*, ao qual é aplicada a seguinte restrição: "...um usuário com perfil Gestor de depósito local tem acesso apenas às informações dos fornecedores da mesma nação que a dele.".

O comando de inserção com os campos é mostrado na Figura 125.

```
insert into supplier
values (s_suppkey, s_name, s_address, s_nationkey, s_phone, s_acctbal,
s_comment);
```

Figura 125 – Comando de inserção na tabela BASE_SUPPLIER

Os comandos executados para realizar a inserção de fornecedores foram os comandos apresentados na Figura 126 (no qual é inserido um fornecedor do Brasil – indicado pela chave de nação 2) e Figura 127 (no qual é inserido um fornecedor da Etiópia – indicado pela chave de nação 5).

```
insert into supplier
values (3000, 'policy_test_1', 's_address', 2, '00000000', 1111111.0,
'testando a política do vpd para inserção');
```

Figura 126 – Comando de inserção de fornecedor do Brasil

```
insert into supplier
values (3001, 'policy_test_2', 's_address', 5, '00000000', 1111111.0,
'testando a política do vpd para inserção');
```

Figura 127 - Comando de inserção de fornecedor da Etiópia

Inicialmente foi realizada a inserção utilizando um usuário sobre o qual não eram aplicadas regras de autorização. Na realidade, utilizamos a base de dados sem regra de autorização cadastrada para a tabela *supplier*. Os resultados são apresentados na Figura 128 (mostrando que foi inserido o registro com *s_name* igual a *'policy_test_1'*) e Figura 129 (mostrando que foi inserido o registro com *s_name* igual a *'policy_test_2'*), respectivamente. Como podemos observar o usuário sem restrições consegue realizar as inserções normalmente.

```
2988                    Supplier#000002988          Ck B7MmscxnrrFaB7H8cLxXUYY,26X
2989                    Supplier#000002989          mHyjXyVDWKtiwQmeEcNXXksmyKrYDC
3000                    policy_test_1               s_address

2938 rows selected
```

Figura 128 - Teste de inserção de fornecedor do Brasil sem controle de acesso

```
2988                    Supplier#000002988          Ck B7MmscxnrrFaB7H8cLxXUYY,26X
2989                    Supplier#000002989          mHyjXyVDWKtiwQmeEcNXXksmyKrYDC
3000                    policy_test_1               s_address
3001                    policy_test_2               s_address

2939 rows selected
```

Figura 129 - Teste de inserção de fornecedor da Etiópia sem controle de acesso

Após este primeiro teste, as linhas inseridas foram removidas, a fim de podermos utilizar os mesmos comandos novamente. Agora, para o segundo teste, foi utilizado, um usuário com as restrições relativas ao perfil *Gestor de Depósito* e da nação Etiópia. O esperado é que o usuário não consiga inserir pelo primeiro comando, mas consiga pelo segundo, contudo, o VPD não teve o funcionamento esperado, o usuário conseguiu inserir uma linha que ele não consegue ver.

Por não possuir cláusula *where,* o comando de inserção deve ser tratado de maneira diferente dos comandos que a possuem. Para que a política funcione corretamente para o comando de inserção é preciso que o parâmetro UPDATE_CHECK seja passado como verdadeiro (*true*) para a função *add_policy*. Se este parâmetro não for declarado na função ele recebe como valor *default* falso, fazendo que a função não seja testada para a inserção.

Após adicionar a política novamente a tabela *suppliers,* agora com o UPDATE_CHECK como verdadeiro, o teste foi repetido e os resultados são mostrados na Figura 130 e na Figura 131. Dessa vez, o usuário com restrições não conseguiu inserir a linha que ele não poderia ver (a do fornecedor do Brasil), o que era o comportamento esperado, contudo, perde-se a transparência do VPD, pois é retornado um erro para o usuário que tentou inserir (Figura 130). Em relação ao teste de inserção de fornecedor da Etiópia, a inserção ocorreu normalmente, pois é da mesma nação que o usuário (Figura 131).

```
Erro ao iniciar na linha 1 no comando
insert into tpch.base_supplier
values (3000, 'policy_test_1', 's_address', 2, '00000000', 1111111.0, 'testa
Relatório de erro:
Erro SQL: ORA-28115: política com violação de opção de verificação
28115. 00000 -  "policy with check option violation"
*Cause:    Policy predicate was evaluated to FALSE with the updated values.
*Action:
```

Figura 130 - Teste de inserção de fornecedor do Brasil com controle de acesso para usuário que não tem perfil para inserir o registro

```
2973                 Supplier#000002973        x95AAV25GHIAYU yhQsM
2979                 Supplier#000002979        1 YrwL2c5fBzytrUl
2985                 Supplier#000002985        XcGr0N6iJDJ4Nki
3001                 policy_test_2             s_address

113 rows selected
```

Figura 131 - Teste de inserção de fornecedor da Etiópia com controle de acesso para usuário que tem perfil para inserir o registro

Observe que o funcionamento do VPD com UPDATE_CHECK para a inserção segue o seguinte padrão:

- Quando definido como *false*
  - O usuário sempre consegue inserir o registro.
- Quando definido como *true*
  - Quando o usuário tenta inserir um registro, o VPD testa se ele terá acesso ao registro:
    - Se o usuário tiver acesso ao registro depois da inserção, ocorre a inserção.
    - Se o usuário não tiver acesso ao registro depois da inserção, levanta a exceção.

7.2.2 Atualização

Antes de serem realizados os testes de atualização foi realizada a consulta apresentada na Figura 132 com um usuário sem nenhuma restrição. O resultado desta consulta é apresentado na Figura 133. Em particular, observe a linha referente ao *supplier* com *s_name* igual a *policy_test_1*. Observe que o atributo *s_phone* deste usuário tem valor '00000000'. Este será o registro utilizado nos testes.

```
select s_suppkey, s_name, s_nationkey, s_phone
from supplier
order by s_suppkey;
```

Figura 132 – Consulta para listar os registors da tabela supplier

```
2987                    Supplier#000002987          22                      32-632-644-8964
2988                    Supplier#000002988          14                      24-952-723-3371
2989                    Supplier#000002989          7                       17-282-348-7573
3000                    policy_test_1               2                       00000000

2938 rows selected
```

Figura 133 - Resultado da consulta Base

Após verificarmos como se encontrava a tabela, realizamos a atualização apresentada na Figura 134, com um usuário sem restrições. O resultado é apresentado na Figura 135.

```
update supplier
set s_phone = '11111111'
where s_suppkey = 3000;
```

Figura 134 – Atualização do *supplier* com *s_phone* '00000000' para '11111111'

```
2987                    Supplier#000002987          22                      32-632-644-8964
2988                    Supplier#000002988          14                      24-952-723-3371
2989                    Supplier#000002989          7                       17-282-348-7573
3000                    policy_test_1               2                       11111111

2938 rows selected
```

Figura 135 - Resultado da atualização sem restrições

Para testar o comportamento do VPD para atualizações foi utilizado o mesmo usuário do teste de inserção, que possui o perfil *Gestor de Depósito*, e é da nação *Etiópia*, cujo valor da coluna *s_nationkey* é igual a 5. Logo, ele não pode realizar a atualização no registro com *s_name* igual a *policy_test_1*, pois este registro é da nação *Brasil* (*s_nationkey* é igual 2).

Antes de realizar o teste com restrições, atualizamos a linha para retornar ao estado inicial, mostrado na Figura 133 (com *s_phone* igual a '00000000'). Na Figura 136, podemos ver o estado da tabela após a tentativa de atualização pelo usuário com restrições, observe que esse resultado é uma consulta realizada pelo usuário sem restrição, afinal o usuário com restrição não poderia ver a entrada da tabela para checar se foi atualizada ou não. É importante ressaltar que, diferentemente da inserção, não foi retornada nenhuma indicação que o comando não foi realizado.

```
2987                    Supplier#000002987          22                      32-632-644-8964
2988                    Supplier#000002988          14                      24-952-723-3371
2989                    Supplier#000002989          7                       17-282-348-7573
3000                    policy_test_1               2                       00000000

2938 rows selected
```

Figura 136 - Resultado da atualização com restrição

Outro teste de atualização interessante de ser realizado é quando o usuário não define uma cláusula *where* no *update*, tentando dessa forma atualizar todos os registros da tabela. Para verificar o comportamento do VPD nesses casos, foi utilizado o comando representada na Figura 137, para o usuário com restrição de acesso.

```
update supplier
set s_phone = '00000000';
```

Figura 137 – Comando de atualização do campo *s_phone* de todos registros da tabela *supplier*

No caso do usuário com restrição, todas as entradas foram atualizadas (Figura 138), afinal, ele atualizou todas as que ele tinha acesso, enquanto que para o usuário sem restrição, somente algumas linhas foram atualizadas (Figura 139).

```
2760                    Supplier#000002760        5                    00000000
2770                    Supplier#000002770        5                    00000000
2822                    Supplier#000002822        5                    00000000
2858                    Supplier#000002858        5                    00000000
2878                    Supplier#000002878        5                    00000000
2925                    Supplier#000002925        5                    00000000
2958                    Supplier#000002958        5                    00000000
2964                    Supplier#000002964        5                    00000000
2973                    Supplier#000002973        5                    00000000
2979                    Supplier#000002979        5                    00000000
2985                    Supplier#000002985        5                    00000000

112 rows selected
```

Figura 138 - Resultado da atualização com restrição e sem cláusula *where*, na visão do usuário com restrição

```
2979                    Supplier#000002979        5                    00000000
2980                    Supplier#000002980        3                    13-121-795-3494
2981                    Supplier#000002981        19                   29-591-615-9173
2982                    Supplier#000002982        1                    11-661-855-9797
2983                    Supplier#000002983        4                    14-331-253-4156
2984                    Supplier#000002984        4                    14-829-961-8448
2985                    Supplier#000002985        5                    00000000
2986                    Supplier#000002986        10                   20-957-807-3635
2987                    Supplier#000002987        22                   32-632-644-8964
2988                    Supplier#000002988        14                   24-952-723-3371
2989                    Supplier#000002989        7                    17-282-348-7573

2937 rows selected
```

Figura 139 - Resultado da atualização com restrição e sem cláusula where, na visão do usuário sem restrição

Assim como na inserção, o funcionamento do VPD em atualizações difere em alguns casos de acordo com a definição do UPDATE_CHECK da política, seguindo o seguinte padrão:

- Quando definido como *false*
    - Quando o usuário tenta atualizar um registro, o VPD testa se ele tem acesso ao registro:
        - Se o usuário não tiver acesso ao registro, não ocorre a atualização, como se o registro não existisse.
        - Se o usuário tiver acesso ao registro, ocorre a atualização, mesmo que essa atualização faça com que o usuário deixe de ter acesso ao registro.

          Exemplo: O usuário tem acesso aos *suppliers* da nação X, ele atualiza a nação de um *supplier*, trocando de X para Y, e consequentemente perdendo o acesso a esse *supplier*.
- Quando definido como *true*

- Quando o usuário tenta atualizar um registro, o VPD testa se ele tem acesso ao registro:
  - Se o usuário não tiver acesso ao registro, não ocorre a atualização, ou seja, como se o registro não existisse.
  - Se o usuário tiver acesso ao registro, o VPD testa se ele terá acesso ao registro depois da atualização:
    - Se o usuário tiver acesso ao registro depois da atualização, ocorre a atualização.
    - Se o usuário não tiver acesso ao registro depois da atualização, levanta exceção.

      Exemplo: O usuário tem acesso aos *suppliers* da nação X, ele tenta atualizar a nação de um *supplier*, trocando de X para Y, e consequentemente perdendo o acesso a esse *supplier,* o VPD levanta exceção, impedindo a atualização.

7.2.3 Remoção

Antes de realizar os testes de remoção foi realizada a consulta apresentada na Figura 140, a fim de saber o estado inicial da tabela onde será testada a remoção. O resultado desta consulta pode ser visto na Figura 141.

```
select s_suppkey, s_name, s_nationkey, s_phone
from supplier
order by s_suppkey;
```

Figura 140 – Consulta para listar os dados existens na tabela antes de realizar a remoção

```
2987                 Supplier#000002987        22                 32-632-644-8964
2988                 Supplier#000002988        14                 24-952-723-3371
2989                 Supplier#000002989        7                  17-282-348-7573
3000                 policy_test_1             2                  00000000

2938 rows selected
```

Figura 141 - Resultado da consulta base

Para testar o funcionando do VPD para remoção, foram realizados realiza dois testes, ambos utilizando o comando apresentado na Figura 142.

```
delete from supplier
where s_suppkey = 3000;
```

Figura 142 – Comando para teste de remoção

No primeiro teste, executamos a remoção com um usuário sem restrições. O resultado esperado é a linha removida. Então, para o segundo teste, a linha será inserida novamente e executaremos novamente o comando de remoção, mas desta vez com um usuário com restrições de autorização.

Para o segundo teste utilizamos o mesmo usuário dos testes de inserção e atualização, que possui o perfil *Gestor de Depósito* e pertencente a nação da Etiópia (*s_nationkey* igual a 5). Novamente o resultado esperado é que o usuário não poderá executar o comando de remoção, pois o fornecedor a ser removido é da nação 2 (Brasil).

Na Figura 143 é mostrada a tabela *supplier* após a execução do comando por um usuário sem restrições. Como pode ser observado o usuário sem restrições consegue remover a linha normalmente.

```
2987                 Supplier#000002987      22                 32-632-644-8964
2988                 Supplier#000002988      14                 24-952-723-3371
2989                 Supplier#000002989      7                  17-282-348-7573

2937 rows selected
```

Figura 143 - Consulta após deletar sem restrição

O resultado do comando de remoção para um usuário com restrições é mostrado na Figura 144. Como pode ser observado, o usuário não conseguiu executar o comando de remoção. Novamente, assim como no caso da atualização, a consulta apresentada na Figura 144 foi feita pelo usuário sem restrição, pois o usuário com restrição não poderia ver a entrada *policy_test_1*, o que poderia passar impressão errada de que ela foi removida. É importante ressaltar que não é retornada nenhuma indicação que o comando não foi realizado.

```
2987                 Supplier#000002987      22                 32-632-644-8964
2988                 Supplier#000002988      14                 24-952-723-3371
2989                 Supplier#000002989      7                  17-282-348-7573
3000                 policy_test_1           2                  00000000

2938 rows selected
```

Figura 144 - Consulta após tentar deletar com restrição

Outro teste de deleção interessante de ser realizado é quando o usuário não define uma cláusula *where* no *delete*, tentando dessa forma deletar todos os registros da tabela. Para verificar o comportamento do VPD nesses casos, foi utilizado o comando representada na Figura 145, tanto para o usuário com restrição de acesso quanto para o usuário sem restrição.

```
delete from supplier
```

Figura 145 - Comando de deleção de todos os registros da tabela *supplier*

O teste foi feito para o usuário com restrição. A Figura 146 apresenta o resultado na visão do usuário com restrição e a Figura 147 apresenta o resultado na visão do usuário sem restrição. Observe que para o usuário com restrição, todas as entradas foram removidas (zero linhas selecionadas), afinal, ele deletou todas as que ele tinha acesso, enquanto para o usuário sem restrição, somente algumas linhas foram removidas (2825 linhas selecionadas em detrimento das 2937 anteriores).

```
S_SUPPKEY            S_NAME                    S_NATIONKEY          S_PHONE
-------------------- ------------------------- -------------------- ---------------

0 rows selected
```

Figura 146 - Resultado da deleção com restrição e sem cláusula where, na visão do usuário com restrição

```
2987                    Supplier#000002987        22                    32-632-644-8964
2988                    Supplier#000002988        14                    24-952-723-3371
2989                    Supplier#000002989        7                     17-282-348-7573

2825 rows selected
```

Figura 147 - Resultado da deleção com restrição e sem cláusula where, na visão do usuário sem restrição

O UPDATE_CHECK não influencia a remoção. Tanto com o valor false como true, o usuário só consegue remover se ele tiver acesso ao registro. Em nenhum dos casos é levantada exceção.

7.2.4 Subconsultas

Subconsultas também podem ser vistas como consultas aninhadas, ou seja, o parâmetro de uma consulta é outra consulta o que cria uma dependência entre elas, além disso, é interessante observar que tanto a consulta principal quanto a consulta interna podem possuir controle de acesso para um usuário específico.

Para esse teste foi utilizada a consulta C2, que determina o fornecedor que deverá ser selecionado para atender uma determinada parte em uma determinada região, utilizando como região 'AFRICA' e como tipo da parte '%BURNISHED%', a consulta resultante é apresentada na Figura 148.

```
select
  s_acctbal,
  s_name,
  n_name,
  p_partkey,
  p_mfgr,
  s_address,
  s_phone,
  s_comment
from
  part,
  supplier,
  partsupp,
  nation,
  region
where
  p_partkey = ps_partkey
  and s_suppkey = ps_suppkey
  and p_size = 48
  and p_type like '%BURNISHED%'
  and s_nationkey = n_nationkey
  and n_regionkey = r_regionkey
  and r_name = 'AFRICA'
  and ps_supplycost = (
```

```
      select
        min(ps_supplycost)
      from
        partsupp, supplier,
        nation, region
      where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'Africa'
      )
order by
  s_acctbal desc,
  n_name,
  s_name,
  p_partkey;
```

Figura 148 – Consulta de teste de comportamento do VPD com subconsultas

Para testar o comportamento do VPD, o usuário deve ter restrições sobre as tabelas relacionadas aos fornecedores (PARTSUPP e SUPPLIER), por essa razão foi utilizado o perfil de acesso *Gestor de depósito* ("... um gestor de depósito só tem acesso aos fornecedores de sua região."), corresponde à proposta de regra T3.

Primeiramente a consulta foi testada com um usuário sem restrição de acesso, o resultado para esse caso pode ser visto na Figura 149, observe que 309 registros foram selecionados e esses registros são de diversas nações diferentes, porém todas da África, como especificado na consulta (Figura 148).

```
-808,16                 Supplier#000005222        MOZAMBIQUE
-808,16                 Supplier#000005222        MOZAMBIQUE
-821,53                 Supplier#000004237        MOZAMBIQUE
-853,92                 Supplier#000006076        ETHIOPIA
-904,63                 Supplier#000004658        ALGERIA
-979,44                 Supplier#000003875        KENYA

309 rows selected
```

Figura 149 - Resultado da consulta C2 sem restrições

A Figura 150 por sua vez apresenta o resultado da consulta utilizando o perfil de acesso *Gestor de depósito* para um usuário da nação *Ethiopia.* Desta vez, somente 67 registros foram retornados, e todos da nação *Ethiopia*. Analogamente, se a região escolhida fosse alguma diferente da África o resultado da consulta seria vazio, o que é exatamente o esperado após a filtragem realizada pelo predicado retornado pelo VPD.

```
-279,72                Supplier#000001450        ETHIOPIA
-415,18                Supplier#000000790        ETHIOPIA
-466,21                Supplier#000003749        ETHIOPIA
-641,69                Supplier#000005970        ETHIOPIA
-727,91                Supplier#000003920        ETHIOPIA
-853,92                Supplier#000006076        ETHIOPIA

67 rows selected
```

Figura 150 - Resultado da consulta C2 com a regra de controle de acesso T3

7.2.5 Group by

A cláusula *group by,* utilizada em funções agregadas, agrupa o resultado de uma consulta em subconjuntos de registros que possuem valores correspondentes em uma ou mais colunas participantes da cláusula *group by*.

Neste teste foi utilizada a consulta C4, que determina o funcionamento do sistema de priorização de pedidos e apresenta uma avaliação da satisfação dos clientes em um determinado período. Neste teste o período selecionado foi '1/1/1989' e '1/1/1999'. A consulta realizada é apresenta na Figura 151.

```
select
  o_orderpriority,
  count(*) as order_count
from orders
where
  o_orderdate >= TO_DATE('1-1-1989')
  and o_orderdate < TO_DATE('1-1-1999') + interval '3' month
  and exists (
      select
        *
      from
        lineitem
      where
        l_orderkey = o_orderkey
        and l_commitdate < l_receiptdate
      )
group by
  o_orderpriority
  order by
  o_orderpriority;
```

Figura 151 – Consulta para teste de *group by*

Para testar o comportamento do VPD, neste caso, o usuário deve ter restrições sobre as tabelas relacionadas às ordens de pedido e aos itens de pedido (ORDERS e LINEITEM, respectivamente), logo foi utilizado o perfil de acesso *Gerente de vendas da Ásia e América* ("Restringir o acesso pelo Gerente de vendas aos itens de pedidos

provenientes de fornecedores das nações do hemisfério Norte das regiões Ásia e América"), detalhado da proposta de regra **Erro! Fonte de referência não encontrada.**.

O primeiro passo foi realizar a consulta utilizando um usuário sem nenhuma restrição de acesso, o resultado é apresentado na Figura 152. Podemos ver que pela utilização do *group by,* a coluna O_ORDERPRIORITY não apresentou nenhum valor repetido, enquanto a coluna ORDER_COUNT apresentou o total de registros agrupados em cada conjunto.

```
O_ORDERPRIORITY ORDER_COUNT
--------------- ----------------------
1-URGENT        179340
2-HIGH          178572
3-MEDIUM        178031
4-NOT SPECIFIED 179114
5-LOW           179069

5 rows selected
```

Figura 152 - Resultado da consulta C4 sem restrições

Em seguida o teste foi repetido utilizando o perfil *Gerente de vendas Ásia e América* para o qual se aplica a regra **Erro! Fonte de referência não encontrada.**. Dessa forma, ao aplicar a regra devem ser considerados apenas os pedidos das regiões Ásia e América dos países do hemisfério norte. O resultado é apresentado na Figura 153. Observe que com a utilização do *group by* mesmo havendo controle de acesso, os mesmos cinco grupos foram retornados, pois existem registros na Ásia e América que são enquadrados nos cinco grupos; contudo, observe que há uma grande diferença no total de registros agrupados em cada conjunto. Esta diferença ocorreu por causa do filtro realizado pela regra de autorização.

```
O_ORDERPRIORITY ORDER_COUNT
--------------- ----------------------
1-URGENT        90263
2-HIGH          89701
3-MEDIUM        89220
4-NOT SPECIFIED 89873
5-LOW           90222

5 rows selected
```

Figura 153 - Resultado da consulta C4 com a regra de controle de acesso T1

7.2.6 Having

A cláusula *having* restringe os resultados do *group by*, de forma parecida como a cláusula *where* age em uma seleção, contudo sendo aplicada a cada grupo da tabela agrupada ou a todo resultado como um único grupo quando o *group by* não for utilizado.

Neste teste foi utilizada a consulta C1, que lista a quantidade de itens que foi vendida, remetida e retornada, incluindo uma cláusula *having sum(l_quantity) < 2800000*. A consulta realizada é apresentada na Figura 154.

```
select
  l_returnflag,
```

```
  l_linestatus,
  sum(l_quantity) as sum_qty,
  sum(l_extendedprice) as sum_base_price,
  sum(l_extendedprice*(1-l_discount)) as sum_disc_price,
  sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as sum_charge,
  avg(l_quantity) as avg_qty,
  avg(l_extendedprice) as avg_price,
  avg(l_discount) as avg_disc,
  count(*) as count_order
from
  lineitem
where
  l_shipdate <= date '1998-12-01' - interval '1' day (3)
group by
  l_returnflag,
  l_linestatus
having
  sum(l_quantity) > 2800000
order by
  l_returnflag,
  l_linestatus;
```

Figura 154 - Consulta para teste de having

Antes de realizar o teste de comportamento com *having* é interessante analisarmos o resultado da consulta sem essa clausula, tanto para o usuário com restrições quanto para o usuário sem restrições. Os resultados das duas consultas podem ser vistos na Figura 155 e na Figura 156, respectivamente.

```
L_RETURNFLAG L_LINESTATUS SUM_QTY              SUM_BASE_PRICE
------------ ------------ -------------------- ----------------------
A            F            2710655              3971449208,01
N            F            69278                101385688,05
N            O            5491890              8058520955,75
R            F            2721084              3987100688,05

4 rows selected
```

Figura 155 - Resultado da consulta sem a clausula having e com controle de acesso

```
L_RETURNFLAG L_LINESTATUS SUM_QTY              SUM_BASE_PRICE
------------ ------------ -------------------- ----------------------
A            F            24520693             35913119182,63
N            F            644878               944968747,5
N            O            49794783             72938782822,09
R            F            24519629             35918879130,3

4 rows selected
```

Figura 156 - Resultado da consulta sem a clausula having e sem controle de acesso

Como já esperado, assim como no teste de comportamento com *group by*, os dois resultados retornaram o mesmo número de linhas, pois formaram o mesmo número de grupos, contudo, podemos perceber, através da soma apresentada na coluna SUM_QTY, que no primeiro caso (com restrição) o usuário tem acesso a menos registros que no segundo caso. Observe agora na Figura 157 e na Figura 158 o que acontece quando incluímos a cláusula *having*. Nesse caso o numero de linhas retornado foi diferente, pois, como pode ser visto no resultado sem a clausula *having*, para o usuário com restrição somente um dos grupos tem a SUM_QTY maior que 2.800.000, enquanto para o usuário sem restrição três grupos têm a SUM_QTY maior que 2.800.000.

```
L_RETURNFLAG L_LINESTATUS SUM_QTY                 SUM_BASE_PRICE
------------ ------------ ---------------------- ----------------------
N            0            5491890                8058520955,75

1 rows selected
```

Figura 157 - Resultado da consulta com a clausula having e com controle de acesso

```
L_RETURNFLAG L_LINESTATUS SUM_QTY                 SUM_BASE_PRICE
------------ ------------ ---------------------- ----------------------
A            F            24520693               35913119182,63
N            0            49794783               72938782822,09
R            F            24519629               35918879130,3

3 rows selected
```

Figura 158 - Resultado da consulta com a clausula having e sem controle de acesso

7.2.7 Merge

O *merge* é uma operação padrão da SQL a partir da revisão de 2003 [Eisenberg *et al.*, 2004], que permite definir ao mesmo tempo atualizações e inserções de um conjunto de informações em uma tabela, como uma espécie de união entre a tabela e as informações de entrada. O *merge* recebe um conjunto de informações e uma cláusula de verificação existência para os registros da tabela que irão receber as informações. A partir dessa cláusula, se o registro existir, é feita uma atualização com as informações de entrada, se o registro não existir é feita uma inserção.

Para realizar esse teste foi escolhida a tabela *supplier* e feita uma cópia dessa tabela chamada de *supplier2*. Além disso, os registros de cada tabela foram removidos de forma a sobrar em cada uma apenas 50 entradas, sendo duas entradas para cada *s_nationkey* diferente, ou seja, cada tabela tem dois fornecedores de cada nação e ainda, um dos fornecedores de cada nação é igual nas duas tabelas e um é diferente. Para poder diferenciar as alterações nas tabelas, em todas as entradas da tabela *supplier* o campo *s_comment* recebeu o valor "teste_comportamento_VPD_merge_A", enquanto na tabela *supplier2* o campo *s_comment* recebeu o valor "teste_comportamento_VPD_merge_B".

O comando utilizado nos testes está representado na Figura 159. Nesse exemplo, a tabela que receberá as novas informações é a tabela *supplier*, e as novas informações estão na tabela *supplier2*. A cláusula de verificação compara o nome dos fornecedores das duas tabelas, caso existam fornecedores com o mesmo nome nas duas tabelas, a entrada do fornecedor na tabela *supplier* será atualizada com o *s_comment* do fornecedor da tabela *supplier2* e, caso exista algum fornecedor na tabela *supplier2* que

não tenha o mesmo nome de algum fornecedor da tabela *supplier*, ele será inserido nela.

```
merge into supplier S1
using supplier2 S2
on (S1.s_name = S2.s_name)
when matched then update set S1.s_comment = S2.s_comment
when not matched then insert values (S2.s_suppkey, S2.s_name,
S2.s_address, S2.s_nationkey, S2.s_phone, S2.s_acctbal, S2.s_comment);
```

Figura 159 - Comando de teste do comportamento do VPD com *merge*

O primeiro teste foi realizado com o usuário sem restrições. O resultado desejado é que metade das entradas já existentes na tabela *supplier* recebam o *s_comment= teste_comportamento_VPD_merge_B* e ainda, 25 novas entradas sejam inseridas, de forma que a tabela fique com 75 entradas, sendo 50 com *s_comment= teste_comportamento_VPD_merge_B* e 25 com *s_comment= teste_comportamento_VPD_merge_A*. Para verificar o resultado foi feita a consulta apresentada na Figura 160, essa consulta conta o número de linhas da tabela *supplier* para cada *s_comment* diferente. O resultado do teste com usuário sem restrições está apresentado na Figura 161. Portanto, o resultado foi exatamente como esperado.

```
select s_comment, count(*) as Total
from supplier
group by s_comment;
```

Figura 160 - Consulta de verificação de resultado do comportamento do VPD com *merge*

```
S_COMMENT                                                                                                    TOTAL
------------------------------------------------------------------------------------------------------------ ----------
teste_comportamento_VPD_merge_A                                                                              25
teste_comportamento_VPD_merge_B                                                                              50

2 rows selected
```

Figura 161 - Resultado do teste de comportamento do VPD com *merge* para o usuário sem restrições

Para o segundo teste, com restrição, foi escolhido um usuário com o perfil *Gestor de depósito local*, que tem acesso à tabela *supplier* somente aos fornecedores da sua nação, no caso a Etiópia (*s_nationkey=5*), enquanto na tabela *supplier2* tem acesso a todas as entradas. O resultado esperado é que somente uma entrada seja atualizada com *s_comment= teste_comportamento_VPD_merge_B* e, também, somente uma nova entrada seja inserida na tabela, resultando em uma tabela com 51 registros, sendo dois com *s_comment= teste_comportamento_VPD_merge_B* e 49 com *s_comment= teste_comportamento_VPD_merge_A*. Além disso, as duas entradas com *s_comment= teste_comportamento_VPD_merge_B* resultantes devem obrigatoriamente ser da nação Etiópia (*s_nationkey=5*), pois são as únicas entradas que o usuário poderia inserir e atualizar. Para verificar isso, além da consulta apresentada na Figura 160, foi elaborada uma nova consulta, apresentada na Figura 162, que verifica o *s_comment* dos fornecedores da Etiópia.

```
select s_nationkey, s_comment
from supplier
where s_nationkey = 5;
```

Figura 162 - Consulta de verificação de resultado do comportamento do VPD com merge pra o usuário com restrição

Quando o teste para o usuário com restrições foi realizado, ocorreu o erro apresentado na Figura 163. O que se descobriu é que o Oracle ainda não suporta o *merge* em tabelas que possuem controle de acesso, na realidade, como não é possível utilizar o merge em conjunto com o VPD, a própria Oracle recomenda que sejam usados comandos insert e update equivalentes para garantir o controle de acesso [ORACLE, 2008].

```
Erro SQL: ORA-28132: A intercalação na sintaxe não suporta políticas de segurança.
28132. 00000 -  "Merge into syntax does not support security policies."
*Cause:    Merge into syntax currently does not support a security policy
           on the destination table.
*Action:   use the insert / update DML stmts on the table that has a
           security policy defined on it.
```

Figura 163 - Erro no teste de comportamento do VPD com merge

Contudo, se o *merge* for feito em uma tabela sem controle de acesso recebendo como entrada uma tabela com controle, a operação ocorre normalmente. Para ilustrar esse comportamento, foi feito um terceiro teste, utilizando o mesmo usuário do segundo teste, e onde a tabela que recebe as informações é agora a *supplier2* e a que fornece as informações é a *supplier*, o comando para esse teste é apresentado na Figura 164.

```
merge into supplier2 S1
using supplier S2
on (S1.s_name = S2.s_name)
when matched then update set S1.s_comment = S2.s_comment
when not matched then insert values (S2.s_suppkey, S2.s_name,
S2.s_address, S2.s_nationkey, S2.s_phone, S2.s_acctbal, S2.s_comment);
```

Figura 164 - Comando de teste do comportamento do VPD com *merge* em uma tabela sem restrição

Analogamente ao resultado esperado no segundo teste, o desejado nesse teste é que somente uma entrada da tabela *supplier2* seja atualizada com *s_comment= teste_comportamento_VPD_merge_A* e, também, somente uma nova entrada seja inserida na tabela, resultando em uma tabela com 51 registros, sendo dois com *s_comment= teste_comportamento_VPD_merge_A* e 49 com *s_comment= teste_comportamento_VPD_merge_B*. Além disso, as duas entradas com *s_comment= teste_comportamento_VPD_merge_A* resultantes devem obrigatoriamente ser da nação Etiópia (*s_nationkey=5*), pois são as únicas entradas que o usuário tem acesso na tabela *supplier*. Para verificar o resultado foram executadas as consultas apresentadas na Figura 160 e na Figura 162, o resultado de cada consulta são apresentados respectivamente na Figura 165 e na Figura 166.

```
S_COMMENT                                                                                          TOTAL
-------------------------------------------------------------------------------------------------- ----------
teste_comportamento_VPD_merge_A                                                                    2
teste_comportamento_VPD_merge_B                                                                    49

2 rows selected
```

Figura 165 - Resultado do teste de comportamento do VPD com merge em uma tabela sem restrições

```
S_NATIONKEY            S_COMMENT
---------------------- ---------------------------------
5                      teste_comportamento_VPD_merge_A
5                      teste_comportamento_VPD_merge_A
5                      teste_comportamento_VPD_merge_B

3 rows selected
```

Figura 166 - Resultado do teste de comportamento do VPD com merge em uma tabela sem restrições

7.2.8 Trigger

A *trigger* é um recurso extremamente poderoso em um SGBD, permitindo a execução de um bloco de comandos a partir da inserção, atualização ou remoção de um ou mais registros em uma tabela, de forma que é possível, por exemplo, programar a atualização de um registro em uma tabela B a partir da atualização de um registro de uma tabela A.

Apesar de muito interessante, essa funcionalidade também é bastante perigosa num cenário de controle de acesso. Imagine, por exemplo, duas tabelas A e B onde B possui uma *foreign key* para A e, além disso, a atualização de um registro na tabela A acarreta, através de uma *trigger*, a atualização de todos os registros de B que apontem para o registro atualizado em A. Imagine também que exista um usuário que tenha acesso a todos os registros de A, mas não tenha acesso a determinados registros de B, o que se deseja testar é: "se o usuário atualiza um registro em A e existe um registro em B onde ele não tem acesso e que aponta para o registro atualizado em A, o registro em B será ou não atualizado pela *trigger*?".

Para realizar o teste de comportamento do VPD, foi criada a *trigger* apresentada na Figura 167. Essa *trigger* define que, sempre que houver uma atualização do campo *n_comment* da tabela *nation*, o campo *s_comment* de todos os registros da tabela *supplier* que apontam para o registro atualizado na tabela *nation* (*s_nationkey* = *n_nationkey*) devem ser atualizados com o mesmo valor.

```
create or replace TRIGGER "TRIGGER_TESTE_COMPORTAMENTO"
after update of n_comment on nation for each row
begin
  update supplier
  set s_comment = :new.n_comment
  where s_nationkey = :new.n_nationkey;
end;
```

Figura 167 - Trigger utilizada para teste de comportamento do VPD

O usuário com restrição utilizado para esse teste possui o perfil *Gestor de depósito*, que limita o acesso apenas aos *suppliers* (tabela SUPPLIER) que possuem a mesma nação que o usuário, que no caso é a Etiópia; contudo, o usuário ainda tem acesso a todas as nações (tabela NATION). O teste foi realizado com o comando apresentado na Figura 168, esse comando atualiza o *n_comment* da nação Estados Unidos (*n_nationkey* = 25) e, conseqüentemente, a *trigger* tentará atualizar o *s_comment* de todos os *suppliers* dos Estados Unidos com o mesmo valor. O resultado esperado é que o VPD impeça que os *suppliers* sejam atualizados, pois o usuário só tem acesso aos *supplies* da Etiópia.

```
update nation
set n_comment = 'teste_comportamento_vpd_trigger'
where n_nationkey = 24;
```

Figura 168 - Comando de atualização para teste de comportamento do VPD com *trigger*

A Figura 169 apresenta o resultado da atualização na tabela *nation* enquanto a Figura 170 apresenta o resultado da atualização na tabela *supplier*. Observe que a consulta utilizada para verificar o resultado da atualização na tabela *suppliers*, foi feita pelo usuário sem restrições, pois o usuário com restrições não poderia selecionar os *suppliers* dos Estados Unidos.

```
22                    RUSSIA                    slowly pending patterns x-ray qu
23                    UNITED KINGDOM            fluffily regular pinto beans bre
24                    UNITED STATES             teste_comportamento_vpd_trigger

25 rows selected
```

Figura 169 - Resultado do teste de atualização com *trigger* na tabela *nation*

```
Supplier#000000201        18                    blithely even theodolites are ir
Supplier#000000151        22                    blithely bold packages according
Supplier#000000202        23                    final, express accounts inte
Supplier#000000152        24                    teste_comportamento_vpd_trigger
Supplier#000001647        1                     carefully even foxes integrate c

2937 rows selected
```

Figura 170 - Resultado do teste de atualização com *trigger* na tabela *supplier*

Como já era esperado, a atualização do campo *n_comment* da nação Estados Unidos foi realizada com sucesso, afinal o usuário tem acesso a todas as nações, contudo, diferentemente do esperado, o *Supplier#000000152* que é da nação Estados Unidos (segunda coluna, *s_nationkey* = 24) também teve o seu campo *s_comment* atualizado. Isso aconteceu por que o usuário dono da *trigger*, que é o mesmo usuário dono das tabelas, possui o privilégio *EXEMPT ACCESS POLICY*, o que faz com que o Oracle ignore as políticas ao executar a *trigger* permitindo dessa forma, que o usuário atualize registros que ele não tenha acesso.

Para evitar esse problema, o privilégio *EXEMPT ACCESS POLICY* foi removido do usuário dono da trigger e o teste foi repetido. O novo resultado da atualização na tabela *nation* está apresentado na Figura 171, e o novo resultado da atualização da tabela *supplier* está apresentado fa Figura 172.

```
22                    RUSSIA                    slowly pending patterns x-ray qu
23                    UNITED KINGDOM            fluffily regular pinto beans bre
24                    UNITED STATES             teste_comportamento_vpd_trigger

25 rows selected
```

Figura 171 - Resultado do teste de atualização com *trigger* na tabela *nation* sem o privilégio *EXEMPT ACCESS POLICY* para o dono da *trigger*

```
Supplier#000000201        18                    blithely even theodolites are ir
Supplier#000000151        22                    blithely bold packages accordinc
Supplier#000000202        23                    final, express accounts inte
Supplier#000000152        24                    hockey players use slyly quickly
Supplier#000001647        1                     carefully even foxes integrate c

2937 rows selected
```

Figura 172 - Resultado do teste de atualização com *trigger* na tabela *supplier* sem o privilégio *EXEMPT ACCESS POLICY* para o dono da *trigger*

Dessa vez, novamente como esperado, o *n_comment* dos Estados Unidos foi atualizado, porém, o VPD impediu que o *s_comment* do *Supplier#000000152* fosse alterado. É importante ainda salientar, que em nenhum momento o usuário é informado que a trigger não consegui realizar a atualização.

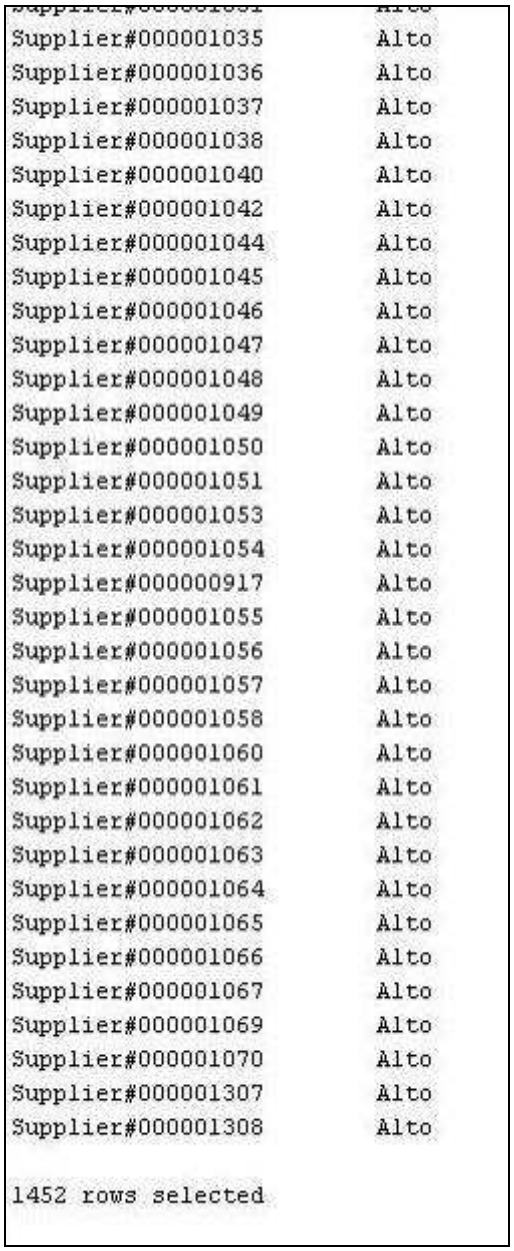
7.2.9 Select case

O comando *SELECT CASE* funciona exatamente como um comando *SWICTH* de linguagem de programação.

Para testar o comando utilizamos duas restrições: uma para a condição e outra para a tabela sobre a qual o comando está sendo executado. Para isto utilizamos a tabela PARTSUPP como condição e a tabela SUPPLIER como tabela base para o comando. A consulta executada é apresentada na Figura 173. Utilizamos essa consulta para saber se o número de pedidos de um fornecedor é maior que 50. O resultado dessa consulta sem nenhuma restrição pode ser visto na Figura 174. Como podemos ver todos os 1452 fornecedores trabalham com mais de 50 peças.

```
select s.s_name,
  case when(( select count(*) from tpch.base_partsupp ps where
ps.ps_suppkey = s.s_suppkey) > 50)
  then 'Alto'
  else 'Baixo'
  end as NUMERO_DE_PEDIDOS
from tpch.base_supplier s;
```

Figura 173 – Consulta para teste do comando *select case*



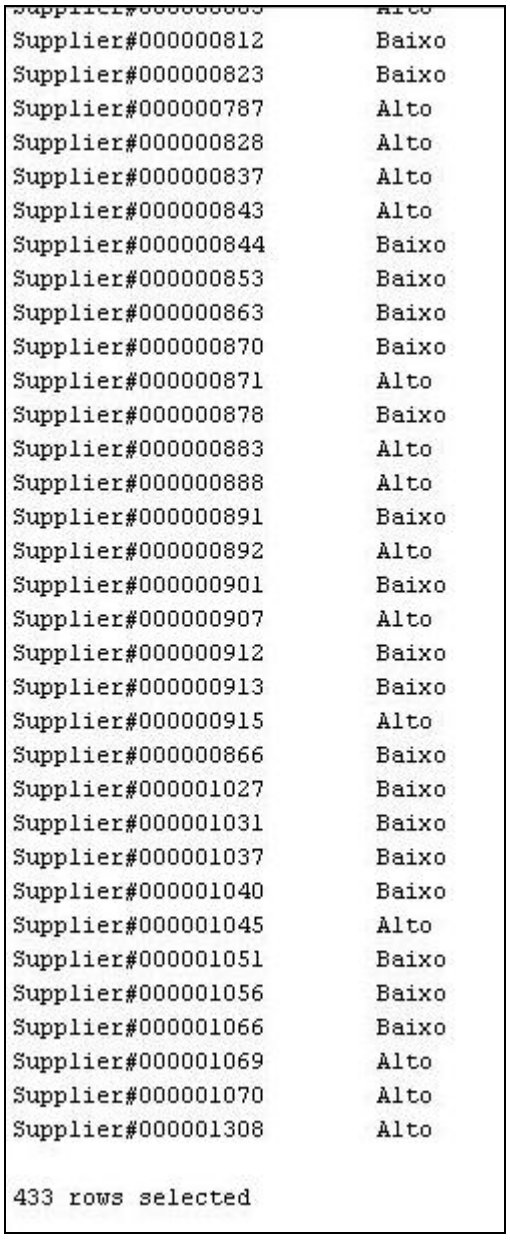
Supplier#000001035	Alto
Supplier#000001036	Alto
Supplier#000001037	Alto
Supplier#000001038	Alto
Supplier#000001040	Alto
Supplier#000001042	Alto
Supplier#000001044	Alto
Supplier#000001045	Alto
Supplier#000001046	Alto
Supplier#000001047	Alto
Supplier#000001048	Alto
Supplier#000001049	Alto
Supplier#000001050	Alto
Supplier#000001051	Alto
Supplier#000001053	Alto
Supplier#000001054	Alto
Supplier#000000917	Alto
Supplier#000001055	Alto
Supplier#000001056	Alto
Supplier#000001057	Alto
Supplier#000001058	Alto
Supplier#000001060	Alto
Supplier#000001061	Alto
Supplier#000001062	Alto
Supplier#000001063	Alto
Supplier#000001064	Alto
Supplier#000001065	Alto
Supplier#000001066	Alto
Supplier#000001067	Alto
Supplier#000001069	Alto
Supplier#000001070	Alto
Supplier#000001307	Alto
Supplier#000001308	Alto

1452 rows selected

Figura 174 - Resultado da Figura 173 sem restrições

Para testarmos o comportamento do VPD sobre este comando utilizamos as seguintes restrições:

Na tabela SUPPLIER o usuário pode somente ver os fornecedores de nações das regiões Ásia e América e que também estejam no hemisfério norte. Sobre a tabela PARTSUPP fizemos a seguinte restrição *PS_AVAILQTY > 3917*, ou seja o usuário consegue somente saber que um fornecedor fornece uma peça se ele possuir mais que 3917 unidades para alguma peça. Com essas restrições realizamos novamente a consulta apresentada na Figura 173. O resultado obtido é apresentado na Figura 175. Agora com as restrições o usuário recebe um resultado diferente. Ele consegue ver somente 433 dos 1452 fornecedores que são os fornecedores que se localizam no hemisfério norte das regiões Ásia ou América e agora os resultados contêm "Alto" e "Baixo". Por exemplo, para o fornecedor *Supplier000000891* que possui 49 peças com mais de 3917 unidades em estoque, a consulta do select case retorna zero, e é retornado *Baixo* para ele.



Supplier#000000812	Baixo
Supplier#000000823	Baixo
Supplier#000000787	Alto
Supplier#000000828	Alto
Supplier#000000837	Alto
Supplier#000000843	Alto
Supplier#000000844	Baixo
Supplier#000000853	Baixo
Supplier#000000863	Baixo
Supplier#000000870	Baixo
Supplier#000000871	Alto
Supplier#000000878	Baixo
Supplier#000000883	Alto
Supplier#000000888	Alto
Supplier#000000891	Baixo
Supplier#000000892	Alto
Supplier#000000901	Baixo
Supplier#000000907	Alto
Supplier#000000912	Baixo
Supplier#000000913	Baixo
Supplier#000000915	Alto
Supplier#000000866	Baixo
Supplier#000001027	Baixo
Supplier#000001031	Baixo
Supplier#000001037	Baixo
Supplier#000001040	Baixo
Supplier#000001045	Alto
Supplier#000001051	Baixo
Supplier#000001056	Baixo
Supplier#000001066	Baixo
Supplier#000001069	Alto
Supplier#000001070	Alto
Supplier#000001308	Alto

433 rows selected

Figura 175 – Resultado da Figura 173 com restrições

Portanto, o VPD executou a restrição na tabela *SUPPLIER* (da cláusula *from*), e para estes registros ele executou a restrição sobre a tabela *PARTSUPP*.

7.2.10 Connect by

O comando *CONNECT BY* é utilizado para tabelas com hierarquia. Esta hierarquia é feita através de um auto-relacionamento.

A sintaxe do comando é mostrada na Figura 176. O comando permite navegar em uma hierarquia definida. Podemos definir também um ponto de início da hierarquia, uma condição de recursividade (geralmente relacionada ao auto-relacionamento).

```
SELECT...
   [START WITH condição_inicial]
   CONNECT BY [sem_ciclos] PRIOR condição_recursiva
   [ORDER SIBLINGS BY cláusula_ordenação]
```

Figura 176 – Sintaxe do comando *Connect by*

Como no *benchmarck* TPC-H não há uma hierarquia definida, para realizarmos o teste para este comando utilizamos o modelo mostrado na Figura 177. O modelo consiste somente de uma tabela. Esta tabela representa uma peça, neste caso, uma peça pode ser composta de várias outras peças, formando assim uma hierarquia.

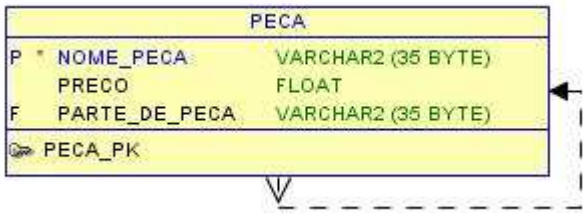


Figura 177 - Modelo para testar Connect By

A Figura 179 mostra o resultado da consulta apresentada na Figura 178 realizada por um usuário sem nenhuma restrição. O campo *level* utilizado na consulta é uma pseudocoluna criada quando é definida a utilização do comando *connect by*, seu valor corresponde ao nível hierárquico de cada registro. Esta coluna é útil, por exemplo, quando de se deseja fazer uma endentação como a realizada na Figura 178.

```
SELECT level, lpad('       ', 8 * (level - 1)) || to_char (nome_peca)
nome_peca, preco
FROM peca
START WITH parte_de_peca IS NULL
CONNECT BY PRIOR nome_peca = parte_de_peca;
```

Figura 178 - Consulta para retornar toda a hierarquia de peças

	LEVEL	NOME_PECA	PRECO
86	5	Correia da ventoinha	30
87	2	Tampa	200
88	1	Sistema Elétrico	135,3
89	2	Buzina	50
90	2	Farol	33
91	3	Lâmpada	20
92	2	Fusível	2
93	1	Transimissão	4900,5
94	2	Câmbio	627
95	3	Alavanca de câmbio	80
96	3	Caixa de câmbio	132
97	4	Polia dentada do comando	80
98	3	Engrenagem	200
99	2	Embreagem	1864,5
100	3	Caixa-seca	100
101	3	Disco de Embreagem	825
102	4	Coroa	100
103	4	Platô de embreagem	300
104	4	Polia	100
105	3	Mola helicoidal	80
106	3	Pedal de Embreagem	50
107	3	Rolamento de Embreagem	100
108	2	Motor de Arranque	300
109	2	Sistema de Transmissão	66
110	3	Eixo cardã	40

Figura 179 - Resultado da consulta de toda a hierarquia

Sobre este esquema foram feitos dois teste para verificar o comportamento do VPD. Ambos os testes foram realizados utilizando a consulta apresentada na Figura 178. No primeiro teste fizemos uma restrição sobre um registro do primeiro nível, no caso "Transmissão" (NOME_PECA<> 'Transmissão'). A Figura 180 mostra o resultado da consulta feita por um usuário com esta restrição. Como podemos perceber o resultado da consulta não mostra nenhum registro que estava abaixo de "Transmissão" como, por exemplo, "Câmbio", "Embreagem" e "Motor de Arranque", ou seja, a hierarquia considerada na restrição não é montada.

	LEVEL	NOME_PECA	PRECO
68	2	Ignição	2772
69	3	Chave de ignição	80
70	3	Silencioso	200
71	3	Vela	429
72	4	Bobina de ignição	200
73	4	Cabo de vela	60
74	3	Válvulas	1270,5
75	4	Balancin	700
76	4	Guia de válvulas	20
77	4	Tampa de válvulas	49,5
78	5	Junta da tampa de válvulas	30
79	2	Radiador	200
80	2	Sistema de Resfriamento	330
81	3	Filtro de ar	173,25
82	4	Entrada do filtro de ar	30
83	4	Feltro	5
84	4	Funil	20
85	4	Ventoinha	49,5
86	5	Correia da ventoinha	30
87	2	Tampa	200
88	1	Sistema Elétrico	135,3
89	2	Buzina	50
90	2	Farol	33
91	3	Lâmpada	20
92	2	Fusível	2

Figura 180 - Resultado da consulta para um usuário com restrição a um registro do nível 1

No segundo teste restringimos um registro pertencente a um nível mais interior da hierarquia ("Disco de embreagem"), a restrição adicionada foi *NOME_PECA* <> *'Disco de Embreagem'*. A Figura 181 mostra o resultado da execução da consulta apresentada na Figura 178 realizada por um usuário com restrições. Como podemos perceber, assim como no teste anterior, não são mostradas as peças que compõe o registro restringido via VPD. Isto pode ser observado comparando o resultado da consulta apresentado na Figura 179 com o resultado da Figura 181. Na Figura 179, abaixo de *Transmissão/Embreagem* é listado o registro *Disco de Embreagem* e todos os seus filhos (*Coroa*, *Platô de Embreagem* e *Polia*). Já com a restrição aplicada (Figura 181), o registro *Disco de Embreagem* e seus filhos não são exibidos.

	LEVEL	NOME_PECA	PRECO
82	4	Entrada do filtro de ar	30
83	4	Feltro	5
84	4	Funil	20
85	4	Ventoinha	49,5
86	5	Correia da ventoinha	30
87	2	Tampa	200
88	1	Sistema Elétrico	135,3
89	2	Buzina	50
90	2	Farol	33
91	3	Lâmpada	20
92	2	Fusível	2
93	1	Transimissão	4900,5
94	2	Câmbio	627
95	3	Alavanca de câmbio	80
96	3	Caixa de câmbio	132
97	4	Polia dentada do comando	80
98	3	Engrenagem	200
99	2	Embreagem	1864,5
100	3	Caixa-seca	100
101	3	Mola helicoidal	80
102	3	Pedal de Embreagem	50
103	3	Rolamento de Embreagem	100
104	2	Motor de Arranque	300
105	2	Sistema de Transmissão	66
106	3	Eixo cardã	40

Figura 181 - Resultado da consulta para um usuário com restrição a um registro interno na hierarquia

Logo, o funcionamento obtido no teste foi exatamente o esperado. Porém se na consulta apresentada na Figura 178 for adicionado um predicado contendo exatamente a mesma condição colocada na política do VPD, como podemos ver na Figura 182, o resultado obtido não é o mesmo obtido utilizando o VPD, como apresentado na Figura 183. Comparando os resultados apresentados na Figura 179 com o resultado apresentado na Figura 183, observe, por exemplo, que, na Figura 179, o registro *Câmbio* está abaixo da hieraquia de *Transmissão*, enquanto que na Figura 183 este registro está abaixo da hierarquia de *Sistema Elétrico*. Esta diferença ocorre porque a filtragem feita pela cláusula WHERE ocorre após a construção da hierarquia, o que é diferente do que desejamos que aconteça, a hierarquia só deveria ser construída após a filtragem.

```
SELECT level ,lpad('      ', 8* (level - 1)) || nome_peca nome_peca,
preco
FROM peca
WHERE nome_peca <> 'Transmissão'
START WITH parte_de_peca IS NULL
CONNECT BY PRIOR nome_peca = parte_de_peca;
```

Figura 182 - Emulando o funcionamento do VPD utilizando cláusula WHERE

	LEVEL	NOME_PECA	PRECO
85	4	Ventoinha	49,5
86	5	Correia da ventoinha	30
87	2	Tampa	200
88	1	Sistema Elétrico	135,3
89	2	Buzina	50
90	2	Farol	33
91	3	Lâmpada	20
92	2	Fusível	2
93	2	Câmbio	627
94	3	Alavanca de câmbio	80
95	3	Caixa de câmbio	132
96	4	Polia dentada do comando	80
97	3	Engrenagem	200
98	2	Embreagem	1864,5
99	3	Caixa-seca	100
100	3	Disco de Embreagem	825
101	4	Coroa	100
102	4	Platô de embreagem	300
103	4	Polia	100
104	3	Mola helicoidal	80
105	3	Pedal de Embreagem	50
106	3	Rolamento de Embreagem	100
107	2	Motor de Arranque	300
108	2	Sistema de Transmissão	66
109	3	Eixo cardã	40

Figura 183 - Resultado da execução da consulta da Figura 182 Erro! Fonte de referência não encontrada.

Por outro lado, se incluirmos o predicado diretamente sobre a tabela *PECA*, como apresentado na Figura 184, obtemos um resultado igual ao conseguido com a consulta da Figura 178 utilizando as restrições do VPD, como podemos observar na Figura 185.

```
SELECT level ,lpad('      ', 8* (level - 1)) || nome_peca nome_peca,
preco
FROM (SELECT * FROM tpch.peca
WHERE tpch.peca.nome_peca <> 'Transmissão')
START WITH parte_de_peca IS NULL
CONNECT BY PRIOR nome_peca = parte_de_peca;
```

Figura 184 - Emulando o funcionamento do VPD utilizando a restrição no from

	LEVEL	NOME_PECA	PRECO
68	2	Ignição	2772
69	3	Chave de ignição	80
70	3	Silencioso	200
71	3	Vela	429
72	4	Bobina de ignição	200
73	4	Cabo de vela	60
74	3	Válvulas	1270,5
75	4	Balancin	700
76	4	Guia de válvulas	20
77	4	Tampa de válvulas	49,5
78	5	Junta da tampa de válvulas	30
79	2	Radiador	200
80	2	Sistema de Resfriamento	330
81	3	Filtro de ar	173,25
82	4	Entrada do filtro de ar	30
83	4	Feltro	5
84	4	Funil	20
85	4	Ventoinha	49,5
86	5	Correia da ventoinha	30
87	2	Tampa	200
88	1	Sistema Elétrico	135,3
89	2	Buzina	50
90	2	Farol	33
91	3	Lâmpada	20
92	2	Fusível	2

Figura 185 - Resultado da execução da consulta da Figura 184

7.3 Oportunidades e Limitações identificadas

Dado os resultados dos testes experimentais realizados, podemos concluir que o VPD constitui-se em uma importante ferramenta para ser utilizada para autorização de acesso. Utilizando o VPD podem ser definidas políticas tanto para consultas sobre os dados como também para realização de operações de modificação dos dados (*insert*, *update* e *delete*). A opção UPDATE_CHECK pode ser utilizada em alguma operações para customizar a validação da execução da operação, para os casos de *insert* e *update*.

Como limitações, observa-se que o VPD não funciona para o comando *merge* nas versões 9i e 10g do Oracle. Além disso, as regras de autorizações devem ser cadastradas em funções PL/SQL, o que pode ser custoso caso sejam muitas regras de autorização. A fim de facilitar o cadastro das regras de autorização, propomos um modelo de autorização, apresentado na seção 5. Dessa forma, é necessário cadastrar apenas uma função de autorização, a qual consulta os dados cadastrados nas tabelas de autorização para montar os predicados de autorização. Isto facilita enormemente o trabalho do responsável pelo cadastro das regras de autorização, pois basta incluir registros nas tabelas de autorização, ao invés de criar uma função para cada autorização, bem como regras podem ser reaproveitadas para definir autorizações sobre diferentes tabelas.

Entretanto, mesmo utilizando o modelo de autorização, o cadastro das regras de autorização depende do responsável pela segurança conhecer as tabelas do banco de dados e saber montar as regras em predicados SQL. Isto não é simples para usuários do negócio. Portanto, é necessário projetar uma interface mais simples para que usuários do negócio sejam capazes de definir as regras sem ter que escrever comandos SQL.

8 Conclusões

Este relatório apresentou o estudo e análise das funcionalidades Label Security e Virtual Private Database para implementação de autorização de acesso no Oracle. Após o estudo das funcionalidades, as quais foram descritas na seção 2, foram realizados vários testes experimentais das mesmas a fim de avaliar o uso em relação aos requisitos de execução de regras de autorização em relação a um banco de dados real.

O benchmark TPC-H foi escolhido para realização dos testes. A abordagem empregada foi de reescrever as regras de forma a representá-las no modelo de dados do TPC-H.

Inicialmente foram realizados testes do Label Security, o qual demonstrou como principais problemas o fato de ser um controle de autorização estático, definindo se o usuário pode ou não acessar um registro inteiro, independente dos valores dos campos do registro. Além disso, a regra de autorização referente Label Security é cadastrada como novas colunas na tabela. Cada regra deve ser uma nova coluna, o que pode aumentar muito o custo de armazenamento.

O próximo teste realizado foi do VPD. Segundo a abordagem de uso do VPD da Oracle, é necessário criar praticamente uma função de autorização para cada tabela que se deseja aplicar uma regra de autorização. Em uma grande empresa como a Petrobras, onde existem muitas regras para serem cadastradas, isto levaria a um número muito grande de funções (functions em PL/SQL) para serem criadas e mantidas. Dessa forma, foi proposto, neste trabalho, um modelo genérico para autorização de acesso, o qual permite criar apenas uma única função PL/SQL que é aplicada para todas as tabelas que se deseja realizar controle de autorização. As regras de autorização são cadastradas nas tabelas geradas segundo este modelo e a função de autorização, quando uma tabela é acessada, lê as regras que se referem ao usuário e monta o predicado a ser utilizado pelo VPD para realizar o filtro.

Dessa forma, as regras de autorização escritas para o TPC-H foram avaliadas segundo o modelo proposto. As seguintes operações foram testadas: inserção, atualização, remoção, subconsultas, group by, having, merge, trigger e select case. Os resultados são resumidos na Tabela 1.

Tabela 1 – Resumo dos resultados de testes do VPD utilizando o modelo proposto

Comando	Observação
Inserção	Necessário ajustar o parâmetro UPDATE_CHECK como verdadeiro na aplicação da política (ADD_POLICY). • UPDATE_CHECK = *false* o O usuário sempre consegue inserir o registro. • UPDATE_CHECK = *true* o Quando o usuário tenta inserir um registro, o VPD testa se ele terá acesso ao registro: ▪ Se usuário tem acesso: ocorre a inserção. ▪ Se usuário não tem acesso: é levantada exceção.

Atualização	UPDATE_CHECK define comportamentos distintos para a atualização: • UPDATE_CHECK = *false* o Se usuário não tem acesso ao registro: não ocorre a atualização. o Se usuário tem acesso ao registro: ocorre a atualização, mesmo que atualização faça com que o usuário deixe de ter acesso ao registro. • UPDATE_CHECK = *true* o Se usuário não tem acesso ao registro: não ocorre a atualização. o Se usuário tem acesso ao registro: ▪ Se usuário tiver acesso ao registro depois da atualização: ocorre a atualização. ▪ Se o usuário não tiver acesso ao registro depois da atualização: é levantada exceção.
Remoção	O valor do UPDATE_CHECK não muda a forma de funcionamento da remoção.Se usuário tem acesso à linha, então ocorre remoção. Se usuário não tem acesso à linha, então a remoção não ocorre e não é levantada exceção.
Subconsultas	
Group by	
Having	
Merge	Erro SQL: ORA-28132: A intercação na sintaxe não suporta polítias de segurança. 28132.0000 - "Merge into syntax does not support security policies" * Cause: Merge into syntax currently does not support a security policy on the destination table. * Action: use the insert / update DML stmts on the table that has a security policy defined on it. Testes foram realizados nas versões 10.2.0.3 e 11.1.0,
Trigger	O usuário dono da trigger não pode possuir o privilégio EXEMPT ACCESS POLICY.
Select case	
Connect by	

Como resultado, foi demonstrado a usabilidade do modelo proposto como uma abordagem genérica para tratar regras de autorização. Os próximos passos incluem pesquisar e desenvolver uma interface que seja simples o suficiente para que usuários do negócio possam cadastrar regras de autorização. Pesquisar uma evolução da função de autorização a fim de que ela passa também ser utilizada para tratar mascaramento de campos, o que não é possível atualmente.

Referências Bibliográficas

BRG, **The Business Rules Group**. Disponível em <http://www.businessrulesgroup.org>. Acessado em 12 Dez. 2009.

CHEN, L. e CRAMPTOM, J. **Set Covering Problems in Role-Based Access Control**. In: Proceedings of 14th European Symposium on Research in Computer Security, Saint-Malo, France, 2009.

DOD **Trusted Computer Security Evaluation Criteria**. Department of Defense, DoD 5200.28-STD, 1983.

DOMINGUES, G.R., CIFERRI, C.D.A., CIFERRI, R.R. **VisualTPCH: Uma Ferramenta para a Geração de Dados Sintéticos para Data Warehouse**. In: Anais da Sessão de Demos do XXIII Simpósio Brasileiro de Banco de Dados, p.31-36, 2008.

EISENBERG, A., MELTON, J., KULKARNI, K., MICHELS, J.E., ZEMKE, F. **SQL:2003 Has Been Published**. SIGMOD Record, 33(1):119-126, 2004.

FERRAIOLO, D.F. e KHUN, D. R. **Role-Based Access Control**. In: 15th National Computer Security Conference, pp. 554—563, Baltimore, MD, 1992.

FERRAIOLO, D.F., SANDHU, R., GAVRILA, S., KUHN, D.R., CHANDRAMOULI, R. **Proposed NIST standard for role-based access control**. ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp. 224—274, 2001.

ORACLE. **Restricting Data Access Using Virtual Private Database**. Disponível em: http://www.oracle.com/technology/obe/obe10gdb/security/vpd/vpd.htm Acesso em: 12 ago. 2009

ORACLE. **Using Oracle Label Security**. Disponível em: http://www.oracle.com/technology/obe/obe10gdb/security/ls/ls.htm Acesso em: 14 ago. 2009

ORACLE. **Oracle Database 10gR2 Security Guide**. Oracle, p.97, 2008.

POESS, M., FLOYD, C. **New TPC Benchmarks for Decision Support and Web Commerce**. SIGMOD Record, 29(4):64-71, 2000.

SANDHU, R.S., COYNE, E.J., FEINSTEIN, H.L., YOUMAN, C.E. **Role-based access control models**. IEEE Computer, vol. 29, no. 2, pp 38-47, 1996.

TPCH, **TPC Benchmark H**. Transaction Processing Perfermance Council. Disponível em http://www.tpc.org/tpch/. Acesso em: 10 set. 2009.

TPCH, **TPC Benchmark H Standard Specification Revision 2.8.0**. Transaction Processing Perfermance Council, 2008. Disponível em http://www.tpc.org/tpch/spec/tpch2.8.0.pdf. Acesso em 10 set. 2009.

YANG, L. **Teaching database security and auditing**. ACM SIGCSE'09, v.1, issue 1, pp. 241—245, 2009.