FUNDAMENTOS DE PERMISSÕES

OBJETIVO

Este documento tem por objetivo esclarecer as dúvidas quanto a proposta de funcionamento das permissões e suas hierarquias entre os objetos e os membros do sistema.

Este documento é parte integrante da Biblioteca Eh System Zero da EH SISTEMAS DE INFORMÁTICA.

NOTA LEGAL

Fundamentos de permissões - Eh System Zero.[odt|pdf]

Esclarecimentos quanto ao sistema de permissões de usuários.

Autor: Erlimar Silva Campos (erlimar.campos@ehsistemas.com.br)

ÍNDICE

INTRODUÇÃO

O sistema tem usuários, o usuário tem um perfil com suas informações pessoais (desta forma na tabela de usuário ficam somente os dados de sua credencial para acesso), cada usuário é pertencente de um grupo principal e pode participar de vários outros grupos adicionais.

Além dos usuários, o sistema tem os "objetos" que são os recursos de sistema administrados por permissões. Juntos (usuários, grupos e objetos) compreendem o sistema de permissões, mas nada é feito diretamente nessas tabelas, antes temos as tabelas que distribuem as permissões propriamente ditas, elas se conectam de uma forma ou de outra as tabelas de usuários, grupos e objetos mencionadas anteriormente, porém, iremos focar somente nas tabelas de distribuição de permissões para esclarecer as dúvidas principais.

TABELAS PRINCIPAIS

Abaixo temos a descrição das tabelas principais, e logo após iremos explorar a lógica que utilizam as mesmas para provê um sistema de permissões, simples, expansível e ao mesmo tempo robusto.

systemzero.member

Coluna	Tipo	Descrição
id	INTEGER	Identificação do membro
user_id	INTEGER	Referência a systemzero.member.id
group_id	INTEGER	Referência a systemzero.group.id

Esta tabela vincula um usuário ou grupo (mas nunca os dois), para que possa participar da administração das permissões.

systemzero.permission

Coluna	Tipo	Descrição
object_id	INTEGER	Referência a systemzero.object
member_id	INTEGER	Referência a systemzero.member
read	CHAR(1)	Permite Leitura/Visualização (T|F)
modify	CHAR(1)	Permite Modificar (T|F)

Coluna	Tipo	Descrição
store	CHAR(1)	Permite Inserção/Inclusão (T|F)
unstore	CHAR(1)	Permite Exclusão/Deleção (T|F)
read_flag	CHAR(1)	Comportamento de Leitura/Visualização com relação a hierarquia superior (A|R)
modify_flag	CHAR(1)	Comportamento de Modificação com relação a hierarquia superior (A|R)
store_flag	CHAR(1)	Comportamento de Inserção/Inclusão com relação a hierarquia superior (A|R)
unstore_flag	CHAR(1)	Comportamento de Exclusão/Deleção com relação a hierarquia superior (A|R)

Esta tabela representa o "centro nervoso" do sistema de permissões, é nela que realmente são definidas as permissões de cada membro (usuário ou grupo) para os objetos do sistema.

Operações permitidas

- READ (Leitura):

  Permite que um determinado recurso possa ser lido, tendo seu conteúdo disponível para visualização.

- MODIFY (Modificar):

  Permite que um determinado recurso possa ser modificado, tendo a possibilidade de alterar seu conteúdo, mas não necessariamente de ler.

- STORE (Inclusão):

  Permite a inclusão de novo conteúdo para um determinado recurso. O mesmo que inserir.

- UNSTORE (Exclusão):

  Permite a exclusão do conteúdo de determinado recurso. O mesmo que delete.

Para cada tipo de operação é possível a definição dos seguintes valores (não é permitido um valor diferente, ou nulo):

- T (de TRUE, Verdadeiro) permite a operação;
- F (de FALSE, Falso) não permite a operação.

A HIERARQUIA DE PERMISSÕES

A princípio, observamos que temos simples objetos cadastrados e para cada usuário ou grupo de usuários é dada ou negada permissão específica para esse objeto. Também vemos que um usuário tem um grupo principal, assim sendo é possível que o usuário tenha suas permissões definidas diretamente para ele (o usuário é um membro), e também herda as permissões definidas para o seu grupo principal (o grupo é um membro). Ainda o usuário também pode participar de outros grupos (grupos adicionais) e assim sendo, ele também herda as permissões definidas para esses grupos.

Sendo assim, observemos um pequeno caso de uso para figurar melhor essa explicação.

- Membro 1 (grupo G1);
- Membro 2 (grupo G2);
- Membro 3 (usuário U1);

  Este usuário (U1) tem como grupo principal "G1", e também participa de "G2" como grupo adicional.

- Objeto 1;
- Objeto 2.

Abaixo temos as permissões definidas entre os três membros e os dois objetos:

MEMBRO	OBJETO	R	M	S	U
Membro 1	*Objeto 1*	T	F	F	F
Membro 1	*Objeto 2*	T	T	F	F
Membro 2	*Objeto 1*	F	F	T	T
Membro 2	***Objeto 2***	F	F	F	T
Membro 3	*Objeto 1*	F	F	F	T
Membro 3	***Objeto 2***	T	T	F	T

Uma visão simplória

Se observarmos singularmente a ligação de um objeto e um membro através da tabela ***systemzero.permission***, podemos determinar o que ele pode fazer no sistema simplesmente observando os valores "R-M-S-U", que correspondem respectivamente a READ, MODIFY, STORE e UNSTORE. Assim tomemos como exemplo o ***"Membro 1"*** relacionado ao ***"Objeto 2"***, podemos dizer que:

- ***Membro 1*** pode Ler e Modificar o ***Objeto 2***, mas não pode incluir nem excluir o mesmo.

Sabemos pois, que "Membro 1" é um grupo, mas nas operações de sistema, esta relação

entre “Membro” e “Objeto” é transparente, e o que será percebido então será a relação entre um “Usuário” e um “Objeto”.

Voltando ao caso de uso apresentado, temos o usuário “U1”, que pertence a dois grupos, “G1” e “G2”. O usuário herda as permissões de ambos os grupos além de ter suas próprias. Assim, em alguns casos teremos conflito, como o que há entre o “Membro 2” e “Membro 3” em relação ao “Objeto 2”.

Observando melhor

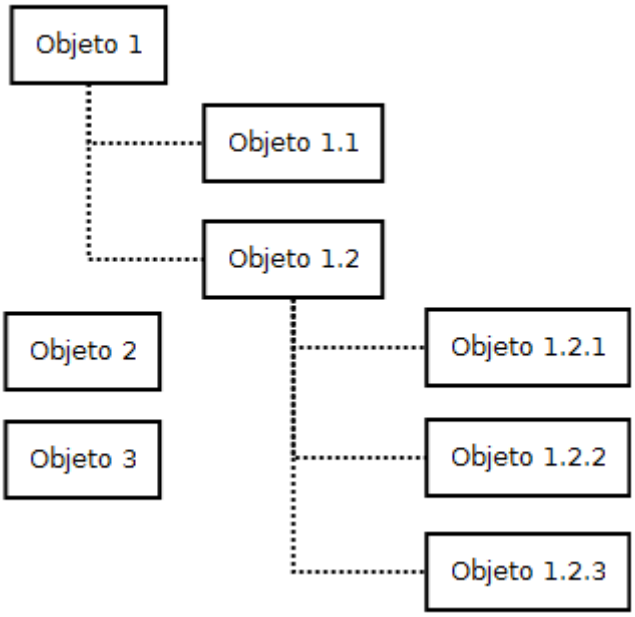
- ***Membro 2*** NÃO PODE Modificar o ***Objeto 2***;
- ***Membro 3*** PODE Modificar o ***Objeto 2***.

Assim, se o usuário “U1” (Membro 3) além de sua permissão (pode modificar), herdou a permissão (não pode modificar) do grupo adicional “G2” (Membro 2), o que podemos determinar? *O usuário **U1** pode, ou não pode, modificar o **Objeto 2**?*

Enquanto essa pergunta se afixa em sua mente, e você pensa um pouco mais para tentar respondê-la, falemos ainda sobre uma outra coisa.

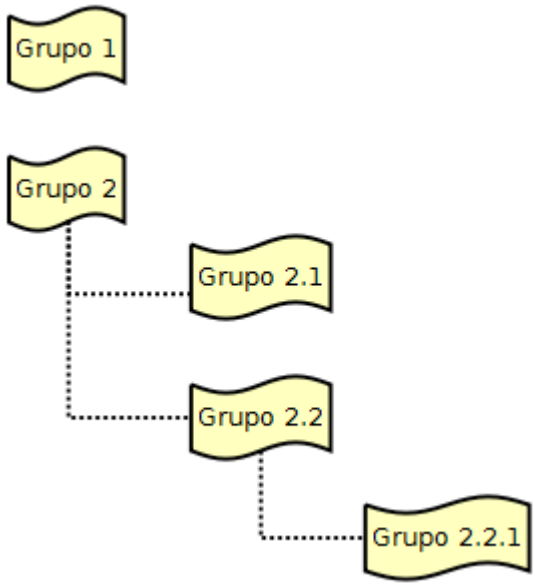
A hierarquia propriamente dita

O título deste capítulo fala sobre “a hierarquia de permissões”, e isto é porque os objetos podem ter subobjetos aninhados abaixo de si, veja a figura abaixo:



1. Figura: Hierarquia de objetos

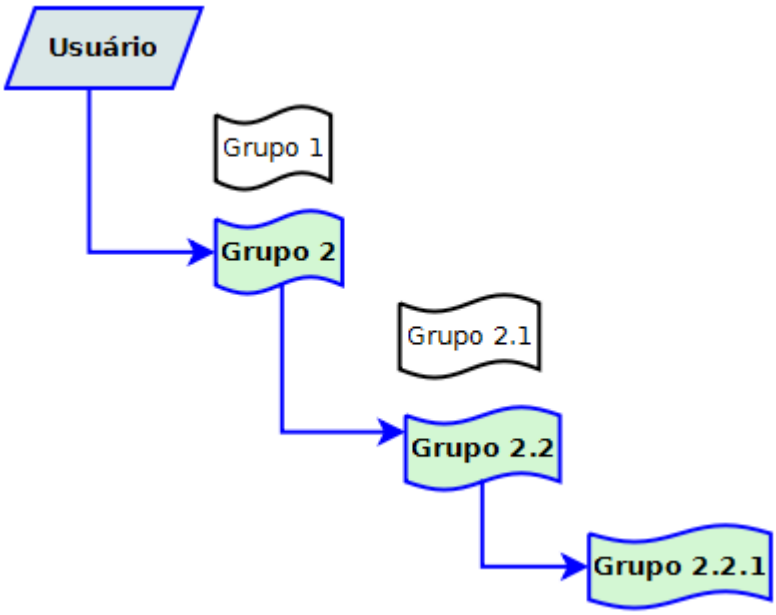
E assim como os objetos, os grupos também podem ter subgrupos aninhados abaixo de si, como na figura abaixo:



2. Figura: Hierarquia de grupos

Somente os usuários não tem 'subusuários' abaixo de si, porém herdam as permissões dos grupos com toda sua hierarquia, e deve ter ou não a permissão sobre os objetos, que também tem uma hierarquia.

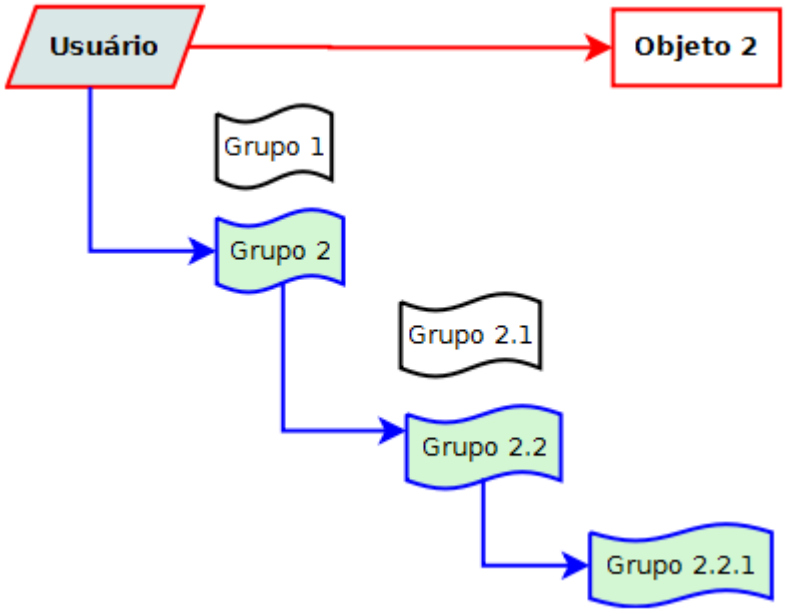
Vale mencionar que os grupos têm um número que os ordena em uma lista de grupos, lista esta que se inicia no próprio usuário e percorre toda a hierarquia até chegar ao grupo específico. Veja a próxima figura:



3. Figura: Percorrendo a hierarquia de permissões

O "Grupo 1" tem número de ordem menor que o número do "Grupo 2" por isso é primeiro na lista, o mesmo acontece com o "Grupo 2.1" em relação ao "Grupo 2.2" e em todos os outros grupos existentes.

Já foi mencionado neste documento que toda a operação de sistema no relacionamento "Membro" e "Objeto" será transparente. Assim, a verificação feita sempre será da permissão do usuário para o objeto. Veja na figura abaixo:



4. Figura: A relação usuário objeto

Podemos ver na figura 4, que é verificada a permissão do "Usuário" para o "Objeto 2". Porém como o usuário é pertencente direta ou adicionalmente aos grupos ilustrados, e os grupos por sua vez também têm permissões (ou negações) para o "Objeto 2", o sistema percorre um caminho síncrono pela hierarquia dos grupos do usuário para definir a real permissão do "Usuário" para o "Objeto 2".

É neste ponto que entra em ação as "flag's" da tabela ***systemzero.permission***, a saber: 'read_flag', 'modify_flag', 'store_flag' e 'unstore_flag', para cada flag é possível a definição dos seguintes valores (não é permitido um valor diferente, ou nulo):

- A (de ADD, Adicionar) adiciona permissão se ainda não existe;
- R (de REPLACE, Substituir) substitui permissão mesmo que já exista.

Juntando as coisas

Na situação proposta temos:

- ***Membro 2*** NÃO PODE Modificar o ***Objeto 2***;
- ***Membro 3*** PODE Modificar o ***Objeto 2***.

Sabendo que "Membro 2" é o "Grupo 2" da figura 4, e "Membro 3" é o "Usuário" da figura 4, então relembremos a pergunta: *"O usuário **U1** pode, ou não pode, modificar o **Objeto 2**?"*.

E a resposta é:

Depende das flag's configuradas no decorrer do caminho percorrido pelo sistema de hierarquias dos grupos.

Então, vamos agora percorrer o caminho do sistema quando lhe é solicitado uma verificação de permissão do "Usuário" para o "Objeto 2", quanto a modificação:

1. O sistema inicia com a permissão NÃO DEFINIDA;
2. O sistema verifica primeiramente a relação do "Usuário" com o "Objeto 2", então encontra que o mesmo PODE MODIFICAR, e como ainda não têm permissão definida, não verifica a FLAG, e define a permissão como "TEM PERMISSÃO";
3. O sistema verifica a relação do próximo nível, no caso: "Grupo 2" com o "Objeto 2", então encontra que o mesmo NÃO PODE MODIFICAR, e como já tem uma permissão definida, verifica a FLAG (e é aqui que a mágica acontece, dependendo do que está definido na FLAG é que se define a permissão). Se assumirmos que a FLAG está definida como "A" (Adicionar), então a permissão será definida como "TEM PERMISSÃO", pois mesmo que "Grupo 2" não tenha permissão, sua FLAG diz que o seu papel é somente "Adicionar" e não "Substituir", assim as coisas ficam como estão. De outra forma, se sua FLAG estivesse definida como "R" (Substituir), neste ponto a permissão seria definida como "NÃO TEM PERMISSÃO".

Por isso a resposta é DEPENTE DO QUE ESTÁ DEFINIDO NA FLAG da permissão.

CONCLUSÃO

Ao fim podemos afirmar que o sistema de permissões proposto pela biblioteca Eh System Zero é simples o bastante para ser representado em uma relação binária entre usuário e objeto, expansível o suficiente para permitir infinitas possibilidades de grupos e subgrupos de usuários e de objetos, formando uma hierarquia que isola ações como inserção, deleção, modificação e visualização. E, robusto porque consegue administrar de forma clara e definitiva os conflitos existentes neste modelo de relacionamento hierárquico.

É sabido da existência de infinitos outros modelos de sistemas de permissões, e que muitos deles são admiráveis, porém também é sabido que muitos complicam bastante as coisas com o intuito de fazer-se incompreendíveis no sentido de arquitetura, com a proposta de que: "sendo complicado de entender, torna-se complicado de burlar, e assim, mais seguro". Porém a biblioteca Eh System Zero tem por principal preocupação, ser clara e eficiente, por isso a decisão de ser acima de tudo, simples, sem, contudo, deixar de ser segura.