8) Trabalhando com Esquemas

8.1) O que são esquemas e sua importância
8.2) Gerenciando esquemas

8.1) O que são esquemas e sua importância

Um agrupamento de bancos de dados do PostgreSQL contém um ou mais bancos de dados com nome. Os usuários e os grupos de usuários são compartilhados por todo o agrupamento, mas nenhum outro dado é compartilhado entre os bancos de dados. Todas as conexões dos clientes com o servidor podem acessar somente os dados de um único banco de dados, àquele que foi especificado no pedido de conexão.

> **Nota:** Os usuários de um agrupamento de bancos de dados não possuem, necessariamente, o privilégio de acessar todos os bancos de dados do agrupamento. O compartilhamento de nomes de usuários significa que não pode haver, em dois bancos de dados do mesmo agrupamento, mais de um usuário com o mesmo nome como, por exemplo, joel; mas o sistema pode ser configurado para permitir que o usuário joel acesse apenas determinados bancos de dados.

Um banco de dados contém um ou mais *esquemas* com nome, os quais por sua vez contêm tabelas. Os esquemas também contêm outros tipos de objetos com nome, incluindo tipos de dado, funções e operadores. O mesmo nome de objeto pode ser utilizado em esquemas diferentes sem conflito; por exemplo, tanto o esquema_1 quanto o meu_esquema podem conter uma tabela chamada minha_tabela. Diferentemente dos bancos de dados, os esquemas não são separados rigidamente: um usuário pode acessar objetos de vários esquemas no banco de dados em que está conectado, caso possua os privilégios necessários para fazê-lo.

Existem diversas razões pelas quais pode-se desejar utilizar esquemas:

- Para permitir vários usuários utilizarem o mesmo banco de dados sem que um interfira com o outro.
- Para organizar objetos do banco de dados em grupos lógicos tornando-os mais gerenciáveis.
- Os aplicativos desenvolvidos por terceiros podem ser colocados em esquemas separados, para não haver colisão com nomes de outros objetos.

Os esquemas são análogos a diretórios no nível do sistema operacional, exceto que os esquemas não podem ser aninhados.

Mais detalhes em: http://pgdocptbr.sourceforge.net/pg80/ddl-schemas.html

http://www.postgresql.org/docs/8.3/interactive/ddl-schemas.html

\dn – visualizar esquemas

Um banco de dados pode conter vários esquemas e dentro de cada um desses podemos criar várias tabelas. Ao invés de criar vários bancos de dados, criamos um e criamos esquemas dentro desse. Isso permite uma maior flexibilidade, pois uma única conexão ao banco permite acessar todos os esquemas e suas tabelas. Portanto devemos planejar bem para saber quantos bancos precisaremos, quantos esquemas em cada banco e quantas tabelas em cada esquema.

Cada banco ao ser criado traz um esquema public, que é onde ficam todas as tabelas, caso não seja criado outro esquema. Este esquema public não é padrão ANSI. Caso se pretenda ao portável devemos excluir este esquema public e criar outros. Por default todos os usuários criados tem privilégio CREATE e USAGE para o esquema public.

É muito útil para estes casos criar um único banco e neste criar vários esquemas, organizados por áreas: pessoal, administracao, contabilidade, engenharia, etc.

Mas e quando uma destas áreas tem outras sub-áreas, como por exemplo a engenharia, que tem reservatórios, obras, custos e cada um destes tem diversas tabelas. O esquema engenharia ficará muito desorganizado. Em termos de organização o ideal seria criar um banco para cada área, engenharia, contabilidade, administração, etc. E para engenharia, por exemplo, criar esquemas para cada subarea, custos, obras, etc. Mas não o ideal em termos de comunicação e acesso entre todos os bancos.

Quando se cria um banco no PostgreSQL, por default, ele cria um esquema público (public) no mesmo e é neste esquema que são criados todos os objetos quando não especificamos o esquema. A este esquema public todos os usuários do banco têm livre acesso, mas aos demais existe a necessidade de se dar permissão para que os mesmos acessem.

"Esquemas são partições lógicas de um banco de dados. São formas de se organizar logicamente um banco de dados em conjuntos de objetos com características em comum sem criar bancos de dados distintos. Por exemplo, podem ser criados esquemas diferentes para dados dos níveis operacional, tático e estratégico de uma empresa, ou ainda esquemas para dados de cada mês de um ano.

Podem ser utilizados também como meios de se estabelecer melhores procedimentos de segurança, com autorizações de acesso feitas por esquema ao invés de serem definidas por objeto do banco de dados.

O PostgreSQL possui um conjunto de esquemas padrão, sendo que a inclusão de objetos do usuário por padrão é feita no esquema PUBLIC. Ao se criar um banco de dados, o banco apresentará inicialmente os seguintes esquemas:

- information_schema – informações sobre funções suportadas pelo banco. Armazena informações

sobre o suporte a SQL, linguagens suportadas e tamanho máximo de variáveis como nome de tabela, identificadores, nomes de colunas, etc.

- pg_catalog – possui centenas de funções e dezenas de tabelas com os metadados do sistema. Guarda informações sobre as tabelas, suas colunas, índices, estatísticas, tablespaces, triggers e demais objetos.

- pg_toast – Informações relativas ao uso de TOAST (The Oversized-Attribute Storage Technique).

- public – Esquema com as tabelas e objetos do usuário.

Exibir a ordem de busca dos Esquemas:

SHOW SEARCH_PATH;

```
postgres=# SHOW SEARCH_PATH;
  search_path
----------------
 "$user",public
```

$user – mostra que será procurado o esquema com o mesmo nome do usuário atual.
public – que será procurado no esquema public.

Alterando a ordem de busca dos Esquemas adicionando o esquema esquema1:

SET SEARCH_PATH TO public, esquema1;

Fonte: http://postgresqlbr.blogspot.com/2007/06/esquemas-no-postgresql.html

Criando Um Esquema
CREATE SCHEMA nomeesquema;

Criando Esquema e tornando um Usuário dono
CREATE SCHEMA nomeesquema AUTHORIZATION nomeusuario;

Removendo privilégios de acesso a usuário em esquema
REVOKE CREATE ON SCHEMA public FROM PUBLIC
Com isso estamos tirando o privilégio de todos os usuários acessarem o esquema public.

Excluindo Um Esquema
DROP SCHEMA nomeesquema;

Aqui, quando o esquema tem tabelas em seu interior, não é possível apagar dessa forma, temos que utilizar:

DROP SCHEMA nomeesquema CASCADE;
Que apaga o esquema e todas as suas tabelas, portanto muito cuidado.
Obs.: O padrão SQL exige que se especifique RESTRICT (default no PostgreSQL) OU CASCADE, mas nenhum SGBD segue esta recomendação.

Obs.: é recomendado ser explícito quanto aos campos a serem retornados, ao invés de usar * para todos, entrar com os nomes de todos os campos. Assim fica mais claro. Além do mais a consulta terá um melhor desempenho.

Acessando Tabelas Em Esquemas
SELECT * FROM nomeesquema.nometabela;

Privilégios Em Esquemas
\dp – visualizar permissões

REVOKE CREATE ON SCHEMA public FROM PUBLIC; - - Remove o privilégio CREATE de todos os usuários.

Obtendo Informações sobre os Esquemas:

\dn

SELECT current_schema();
SELECT current_schemas(true);
SELECT current_schemas(false);

Schemas ou Databases?

Fabio Telles - na lista de postgresql-br

Realmente é difícil justificar o uso de bancos de dados separados. Em geral, utilizar schemas é sempre mais indicado. Vale a pena lembrar que existem 3 níveis de compartilhamento num mesmo servidor:

* Cluster, que compartilham a mesma porta, os mesmos processos e a mesma configuração global (postgresql.conf e pg_hba.conf);

* Bancos de Dados, que compartilham o mesmo cluster mas podem ter codificação de caracteres distintos e dependendo da configuração, podem ter usuários distintos (se 'db_user_namespace' for configurado como ON);
* Shemas, que compartilham o mesmo banco de dados;

Em geral, utilizar mais de um banco de dados pode ser uma boa se:
- Você realmente precisa ter usuários com poderes plenos num banco de dados sem afetar outras aplicações. Isto é muito incomum, pois é normal você dar permissões para um usuário em um schema específico.
Mas se você quiser soltar seu estagiário para brincar no seu servidor... esta pode ser uma opção (mas eu instalaria o PostgreSQL localmente na máquina dele)
- Você precisa de ambientes de teste, homologação e/ou produção na mesma máquina. Em geral é melhor deixar seu ambiente de teste em outro servidor. Você vai descobrir que uma única consulta mal feita no ambiente de teste pode sentar todo o servidor... melhor evitar isso a todo o custo! Você também pode criar ambientes isolados em clusters separados e garantir mais recursos para o ambiente de testes ao ambiente de produção e ainda deixar os ambientes mais isolados, utilizando portas distintas.
- Você precisa de bancos de dados com codificações de caracteres diferentes. Bom... seria ideal você pensar em utilizar utf-8 para todas as bases se você está nesta situação. Mas isto é uma looooonga
conversa, para lá de complicada.
- Você precisa de configurações de desempenho diferentes para diferentes aplicações (OLTP x BI por exemplo). Bom, você pode fazer isso com schemas também... especificando parâmetros por usuário, ao invés de parâmetros por banco de dados, mas se você quer levar isto realmente a sério, é melhor criar clusters distintos e melhor ainda, utilizar servidores distintos.

Conclusão:
Se mais de uma aplicação podem conviver no mesmo servidor e no mesmo clustrer, então elas podem estar no mesmo banco de dados. Existem raras casos em que isto não se aplique... mas são exceções e não a regra.

Criei um pequeno tutorial explicando como unificar vários bancos de dados distintos utilizando schemas... vide:
http://www.midstorm.org/~telles/2006/09/28/unificando-bases-de-dados-com-schemas/

Atenciosamente,
Fábio Telles

O ideal seria realmente o uso de schemas e centralização de informação. Se você mantiver diversos banco de dados, para cada bando terá que abrir uma conexão, ou então ficar mudando o tempo todo de banco em uso nas suas queries, o que pode causar facilmente erros. Se você tem tabelas compartilhadas entre os sistemas, melhor centralizar tudo mesmo. Aí, você pode aumentar o número de conexões simultâneas sem maiores problemas.

Hoje, onde trabalho, tenho os banco de dados divididos em schemas.

Por exemplo.

O schema security tem tabelas relacionados com segurança como menus dos usuários, direito de cada módulo para cada usuário.

No schema scp temos o sistema de controle de processos.
No schema comum, temos tabelas que são utilizados por qualquer sistema ou schema, como por exemplo o cadastro de usuários que acessam o sistema.

Quando você coloca em banco de dados separado, você pode fazer views para fazer select com junções entre tabelas de banco de dados distintos, mas para isso vc tem que compilar e instalar o módulo dblink e utilizá-lo sempre que efetuar consultas com bancos distintos.

O schema facilita muito a criação de views e selects de tabelas distribuidas.

Quanto ao desempenho, bem, não notei redução de desempenho em nenhum serviço rodando nas 350 estações.

Uma dica. O banco de dados depende muito das operações de leitura e escrita. O mais importante no servidor é ter um bom disco e a configuração do postgresql.conf bem afinada.

Se você ver que o seu disco está sendo muito utilizado e o servidor está meio lento, você pode dividir as tabelas ou até mesmo os schemas em tablespaces. Isso permite vc colocar tabelas em um segundo disco. Quando você faz isso, o planejador pode trabalhar de forma a dividir o trabalho dos discos. Por exemplo:
Você tem os schemas A, B e C. Cada schema tem seu próprio sistema e cada sistema tem em média 100 usuários conectado simultaneamente.
Se você colocar em 3 discos e colocar cada schema em um disco, quando o usuário do sistema A, B e C gravarem dados simultâneamente o postgres vai gravar esses dados mais rápido. Isso porque cada transação é uma thread e as threads vão trabalhar em paralelo gravando cada um em um disco ao invés de uma thread esperar os dados do usuário A gravar para começar a gravar os dados do usuário B.

Cleberson Costa Silva.

Se você tem cópias de tabelas mantidas em seus diversos bancos de dados e necessita manter a integridade entre elas então é melhor utilizar esquemas. Se seus bancos de dados são totalmente independentes então é melhor mantê-los separados.
Analise sob o aspecto da integridade dos dados e seu peso nos sistemas.