PostgreSQL

Manipulando o Banco de Dados

Aula 03

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

Acessando um banco de dados

Após o banco de dados ter sido criado, pode ser acessado pela:

1) Execução do programa de terminal interativo do PostgreSQL chamado psql, que permite a entrada, edição e execução de comandos SQL interativos.

2) Utilização de uma ferramenta cliente gráfica existente como o PgAdmin, ou de um pacote de automação de escritórios com suporte a ODBC para criar e manusear bancos de dados.

3) Criação de aplicações personalizadas, usando uma das várias ligações com linguagens disponíveis, como por exemplo o PHP.

Antes de usarmos o terminal interativo PSQL, vamos ter que criar uma nova base de dados, utilizando o comando createdb no sistema operacional, ou usar o comando create database de dentro do psql.

createdb –U postgres –h 127.0.0.1 nomebancodedados

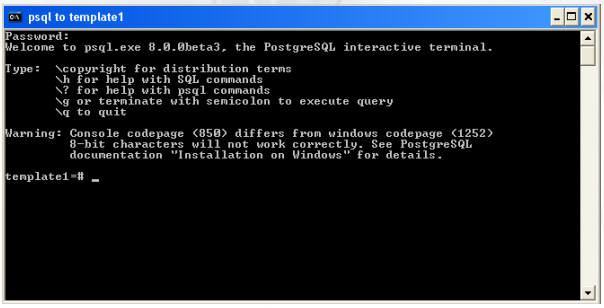
PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Utilizando o PSQL

O psql é um cliente no modo terminal do PostgreSQL. Permite digitar comandos interativamente, submetê-los para o PostgreSQL e ver os resultados. Como alternativa, a entrada pode vir de um arquivo. Além disso, possui vários meta-comandos e diversas funcionalidades semelhantes às da shell para facilitar a criação de scripts e automatizar um grande número de tarefas.



Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-a ou --echo-all
Exibe todas as linhas na tela ao serem lidas. É mais útil para o processamento de scripts que no modo interativo. Equivale a definir a variável ECHO como all.

-A ou --no-align
Muda para o modo de saída não alinhado (De outra forma, o modo de saída padrão é o alinhado).

-c comando ou --command comando
Especifica que o psql deve executar a cadeia de caracteres comando e, em seguida, terminar. Útil em scripts. O comando deve ser uma cadeia de caracteres que possa ser integralmente analisada pelo servidor (ou seja, não contém funcionalidades específicas do psql), ou ser um único comando de contrabarra.
Portanto, não podem ser misturados comandos SQL com meta-comandos do psql. Para misturar, a cadeia de caracteres pode ser enviada para o psql conforme mostrado a seguir: echo "\x \\ select * from foo;" | psql. Se a cadeia de caracteres do comando contém vários comandos SQL, estes são processados em uma única transação, a menos que existam comandos BEGIN/COMMIT explícitos incluídos na cadeia de caracteres para dividi-la em várias transações. Este comportamento é diferente do comportamento ocorrido quando a mesma cadeia de caracteres é recebida pela entrada padrão do psql.

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-d nome_bd ou --dbname nome_bd
Especifica o nome do banco de dados a se conectar. Equivale a especificar nome_bd como o primeiro argumento não-opção na linha de comando.

-e ou --echo-queries
Mostra todos os comandos enviados para o servidor. Equivale a definir a variável ECHO como queries.

-E ou --echo-hidden
Exibe o comando verdadeiro gerado por \d e por outros comandos de contrabarra. Pode ser usado caso se deseje incluir uma funcionalidade semelhante nos próprios programas. Equivale a definir a variável ECHO_HIDDEN a partir do psql.

-f nome_do_arquivo ou --file nome_do_arquivo
Usa o arquivo nome_do_arquivo como origem dos comandos, em vez de ler os comandos interativamente. Após o arquivo ser processado, o psql termina. Sob muitos aspectos equivale ao comando interno \i. Se o nome_do_arquivo for - (hífen) a entrada padrão é lida. O uso desta opção é quase imperceptivelmente diferente de escrever psql < nome_do_arquivo. De uma maneira geral, as duas formas fazem o esperado, mas o uso de -f ativa algumas funcionalidades úteis, como mensagens de erro com o número da linha. Ao se usar esta opção existe, também, uma pequenachance de reduzir o trabalho extra de inicialização. Por outro lado, a utilização do redirecionamento da entrada (em teoria) garante produzir exatamente a mesma saída produzida quando tudo é digitado manualmente.

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-F separador ou --field-separator separador
Usa separador como o separador de campos. Equivale a \pset fieldsep ou ao comando \f.

-h hospedeiro ou --host hospedeiro
Especifica o nome de hospedeiro da máquina onde o servidor está executando. Se o nome iniciar por uma barra (/) é usado como o diretório do soquete do domínio Unix.

-H ou --html
Ativa a saída tabular HTML. Equivale a \pset format html ou ao comando \H.

-l ou --list
Mostra todos os bancos de dados disponíveis e depois termina. Outras opções que não sejam de conexão são ignoradas. Semelhante ao comando interno \list.

-o nome_do_arquivo ou --output nome_do_arquivo
Envia a saída de todos os comandos para o arquivo nome_do_arquivo. Equivale ao comando \o.

-p porta ou --port porta
Especifica a porta TCP, ou a extensão de arquivo do soquete do domínio Unix local, onde o servidor está ouvindo as conexões. Por padrão o valor da variável de ambiente PGPORT ou, se não estiver definida, a porta especificada durante a compilação, geralmente 5432.

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-P atribuição ou --pset atribuição
Permite especificar opções de exibição no estilo \pset pela linha de comando. Observe que aqui o nome e o valor devem estar separados pelo sinal de igual, em vez de espaço. Portanto, para definir o formato de saída como LaTeX deve ser escrito -P format=latex.

-q ou --quiet
Especifica que o psql deve trabalhar em silêncio. Por padrão, são exibidas mensagens de boas-vindas e várias outras mensagens informativas. Se esta opção for usada, nada disso acontece. É útil em conjunto com a opção -c. Dentro do psql é possível, também, definir a variável QUIET para produzir o mesmo efeito.

-R separador ou --record-separator separador
Usa separador como o separador de registros. Equivale ao comando \pset recordsep.

-s ou --single-step
Executa no modo passo-único, significando que será solicitada uma confirmação antes de cada comando ser enviado para o servidor, com a opção de cancelar a execução. Usado para depurar scripts.

-S ou --single-line
Executa no modo linha-única, onde o caractere de nova-linha termina o comando SQL, como o ponto-e-vírgula faz.

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-t ou --tuples-only
Desativa a exibição dos nomes das colunas, do rodapé com o número de linhas do resultado, etc. É inteiramente equivalente ao meta-comando \t.

-T opções_de_tabela ou --table-attr opções_de_tabela
Permite especificar opções a serem colocadas dentro da marca table do HTML. Veja \pset para obter detalhes.

-u
Faz o psql solicitar o nome do usuário e a senha antes de conectar ao banco de dados. Esta opção está obsoleta, porque é conceitualmente incorreto (Solicitar um nome de usuário não padrão e solicitar uma senha porque o servidor requer são realmente duas coisas diferentes). Encoraja-se o uso das opções -U e -W em seu lugar.

-U nome_do_usuário ou --username nome_do_usuário
Conecta ao banco de dados como o usuário nome_do_usuário em vez do padrão (É necessário ter permissão para fazê-lo, é claro).

-v atribuição ou --set atribuição ou --variable atribuição
Realiza atribuição de variável, como o comando interno \set. Observe que é necessário separar o nome e o valor, se houver, por um sinal de igual na linha de comando. Para remover a definição de uma variável omite-se o sinal de igual. Para definir uma variável sem um valor, usa-se o sinal de igual mas omite-se o valor.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

-V ou --version
Mostra a versão do psql.

-W ou --password
Requer que o psql solicite a senha antes de conectar ao banco de dados. Esta continuará definida por toda a sessão, mesmo que a conexão ao banco de dados seja mudada com o meta-comando \connect. Na versão corrente, o psql automaticamente solicita a senha sempre que o servidor requer autenticação por senha. Como atualmente isto se baseia em um hack, o reconhecimento automático pode falhar misteriosamente e, por isso, existe esta opção para forçar a solicitação. Se a solicitação da senha não for feita, e o servidor requer autenticação por senha, a tentativa de conexão não será bem-sucedida.

-x ou --expanded
Ariva o modo formatação de tabela estendido Equivale ao comando \x.

-X ou --no-psqlrc
Não lê o arquivo de inicialização ~/.psqlrc.

-? ou --help
Mostra a ajuda sobre os argumentos de linha de comando do psql.
Status de saída

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

O psql retorna: 0 se terminar normalmente; 1 se ocorrer erro fatal próprio (falta de memória, arquivo não encontrado); 2 se a conexão com o servidor teve problema e a sessão não é interativa; 3 se ocorrer erro no script e a variável ON_ERROR_STOP estiver definida.

O psql é uma aplicação cliente do PostgreSQL comum. Para conectar a um banco de dados é necessário saber o nome do banco de dados, o nome do hospedeiro, o número da porta do servidor e o nome do usuário a ser usado para conectar. O psql pode ser informado sobre estes parâmetros por meio das opções de linha de comando -d, -h, -p e -U, respectivamente. Se for encontrado um argumento que não pertença a nenhuma opção, este será interpretado como o nome do banco de dados (ou o nome do usuário, se o nome do banco de dados já tiver sido fornecido). Nem todas estas opções são necessárias, os padrões se aplicam. Se for omitido o nome do hospedeiro, então o psql se conecta através do soquete do domínio Unix ao servidor no hospedeiro local. O número padrão para a porta é determinado na compilação. Uma vez que o servidor de banco de dados usa o mesmo padrão, não é necessário especificar a porta na maioria dos casos. O nome de usuário padrão é o nome do usuário do Unix, como também é o nome do banco de dados padrão. Observe que não é possível se conectar a qualquer banco de dados com qualquer nome de usuário. O administrador de banco de dados deve informar as permissões de acesso concedidas. Para evitar a digitação, podem ser definidas as variáveis de ambiente PGDATABASE, PGHOST, PGPORT e PGUSER com os valores apropriados.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

Se a conexão não puder ser estabelecida por algum motivo (por exemplo, privilégios insuficientes, o servidor não está executando no hospedeiro, etc.), o psql retorna uma mensagem de erro e termina.

Entrando com comandos SQL

No modo normal de operação, o psql exibe um prompt com o nome do banco de dados ao qual está conectado, seguido pela cadeia de caracteres =>. Por exemplo:

```
$ psql testedb
Welcome to psql 7.4.5, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit
testedb=>
```

No prompt o usuário pode digitar comandos SQL. Normalmente, as linhas de entrada são enviadas para o servidor quando o caractere ponto-e-vírgula, que termina o comando, é encontrado. Um caractere de fim-de-linha não termina o comando. Portanto, os comandos podem ser distribuídos por várias linhas para clareza. Se o comando for enviado e executado sem erro, o resultado do comando será exibido na tela.

Aula 03

PostgreSQL

Terminal Interativo – PSQL - Opções

Meta-comandos

Qualquer texto digitado no psql começando por uma contrabarra (\) (não entre apóstrofos, ') é um meta-comando do psql processado pelo próprio psql. Estes comandos tornam o psql interessante para administração e para scripts. Os meta-comandos são geralmente chamados de comandos de barra ou de contrabarra.

O formato de um comando psql é a contrabarra, seguida imediatamente por um verbo comando, e depois pelos argumentos. Os argumentos são separados do verbo comando, e entre si, por qualquer número de caracteres de espaço.

Para incluir caracteres de espaço no argumento deve-se colocá-los entre apóstrofos ('). Para incluir um apóstrofo neste tipo de argumento, deve-se precedê-lo por uma contrabarra. Qualquer texto entre apóstrofos está sujeito às substituições no estilo C para \n (nova-inha), \t (tabulação), \dígitos, \0dígitos e \0xdígitos (o caractere com o código decimal, octal ou hexadecimal especificado).

Se um argumento (não entre apóstrofos) começar por dois-pontos (:), será considerado como sendo uma variável e o valor desta variável será usado como o argumento.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

Os seguintes meta-comandos estão definidos:

\a
Se o formato atual de saída da tabela for não alinhado, muda para alinhado. Se não for não alinhado, define como não alinhado. Este comando é mantido para compatibilidade com as versões anteriores. Veja \pset para uma solução geral.

\cd [diretório]
Muda o diretório de trabalho corrente para diretório. Sem argumento, muda para o diretório home do usuário corrente. Dica: Para ver o diretório de trabalho corrente deve ser usado \!pwd.

\C [título]
Define o título de qualquer tabela mostrada como resultado de uma consulta, ou remove a definição deste título. Este comando equivale a \pset titletítulo (O nome deste comando deriva de "caption" (título), porque anteriormente só era usado para definir título em tabela no HTML).

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\connect (ou \c) [nome_bd [nome_do_usuário]]
Estabelece a conexão com um banco de dados novo e/ou sob um usuário novo. A conexão anterior é fechada. Se nome_bd for - (hífen), então é utilizado o nome do banco de dados corrente. Se o nome_do_usuário for omitido, então o nome do usuário corrente é utilizado. Como regra especial, \connect sem nenhum argumento conecta ao banco de dados padrão com o usuário padrão (da mesma forma que aconteceria se o psql fosse executado sem argumentos). Se a tentativa de conexão não for bem-sucedida (nome de usuário errado, acesso negado, etc.), a conexão anterior será mantida se, e somente se, o psql estiver no modo interativo. Ao executar um script não interativo, o processamento será imediatamente interrompido com erro. Esta distinção foi escolhida por ser mais conveniente para o usuário que digita menos, e como um mecanismo de segurança impedindo os scripts atuarem acidentalmente no banco de dados errado.

\copy tabela [(lista_de_colunas)] { from | to } nome_do_arquivo | stdin | stdout [with] [oids] [delimiter [as] 'caractere'] [null [as] 'cadeia_de_caracteres']
Executa uma cópia pelo cliente. Esta é uma operação que executa o comando SQL COPY, mas em vez do servidor ler ou escrever no arquivo especificado, o psql lê ou escreve no arquivo e roteia os dados entre o servidor e o sistema de arquivos local. Isto significa que a acessibilidade ao arquivo e os privilégios são do usuário local, e não do servidor, e que não há necessidade dos privilégios do superusuário. A sintaxe deste comando é semelhante à sintaxe do comando SQL COPY. Observe que, por isso, regras especiais de análise se aplicam ao comando \copy. Em particular, as regras de substituição de variável e de escapes de contrabarra (\) não se aplicam.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\copyright
Mostra os termos da distribuição e dos direitos autorais do PostgreSQL.

\d [padrão]
Para cada relação (tabela, visão, índice ou seqüência) correspondendo ao padrão, mostra todas as colunas, seus tipos e atributos especiais como NOT NULL ou o valor padrão, se houver. Os índices, restrições, regras e gatilhos associados também são mostrados, assim como a definição da visão se a relação for uma visão (A "Correspondência com padrão" é definida abaixo). A forma \d+ do comando é idêntica, mas todos os comentários associados às colunas da tabela também são mostrados.
Nota: Se \d for utilizado sem o argumento padrão, torna-se equivalente a \dtvs, mostrando a lista de todas as tabelas, visões e seqüências. Isto é puramente uma medida de conveniência.

\da [padrão]
Mostra todas as funções de agregação disponíveis, junto com o tipo de dado com que operam. Se padrão for especificado, somente as agregações cujos nomes correspondem ao padrão são mostradas.

\dc [padrão]
Mostra todas as conversões entre codificações de conjunto de caracteres disponíveis. Se padrão for especificado, somente as conversões cujos nomes correspondem ao padrão são mostradas.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\dC
Mostra todas as conversões de tipo disponíveis.

\dd [padrão]
Mostra a descrição dos objetos cujos nomes correspondem ao padrão, ou todos os objetos visíveis se nenhum argumento for especificado. Nos dois casos somente os objetos que possuem uma descrição são mostrados ("Objeto" compreende agregações, funções, operadores, tipos, relações [tabelas, visões, índices, seqüências e objetos grandes], regras e gatilhos). Por exemplo:

```
=> \dd version
                  Object descriptions
   Schema   | Name   | Object |        Description
------------+---------+----------+---------------------------
 pg_catalog | version | function | PostgreSQL version string
(1 linha)
```

As descrições dos objetos podem ser criadas pelo comando SQL COMMENT ON.

\dD [padrão]
Mostra todos os domínios disponíveis. Se padrão for especificado, somente os domínios cujos nomes correspondem ao padrão são mostrados.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\df [padrão]
Mostra as funções disponíveis, junto com o tipo de dado de seus argumentos e do valor retornado. Se padrão for especificado, somente as funções cujos nomes correspondem ao padrão são mostradas. Se a forma \df+ for usada, informações adicionais sobre cada função, incluindo a linguagem e a descrição, são mostradas.
Nota: Para diminuir a confusão, \df não mostra as funções com tipo de dado I/O. Isto é implementado ignorando as funções que recebem ou retornam o tipo cstring.

\distvS [padrão]
Este não é na verdade o nome do comando: As letras i, s, t, v, S correspondem a índice, seqüência, tabela, visão e tabela do sistema, respectivamente. Pode ser especificada qualquer uma ou todas as letras, em qualquer ordem, para obter a relação de todos os objetos correspondentes. A letra S restringe a listagem aos objetos do sistema; sem o S, somente os objetos que não são do sistema são mostrados. Se o caractere "+" for anexado ao nome do comando, cada objeto é mostrado junto com sua descrição associada, se houver. Se o padrão for especificado, somente os objetos cujos nomes correspondem ao padrão são mostrados.

\dl
Este é um outro nome para \lo_list, que mostra a lista dos objetos grandes.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\dn [padrão]
Mostra todos os esquemas (espaços de nomes) disponíveis. Se padrão (uma expressão regular) for especificado, somente os esquemas cujos nomes correspondem ao padrão são mostrados.

\do [padrão]
Mostra os operadores disponíveis junto com o tipo de seus operandos e do valor retornado. Se padrão for especificado, somente os operadores cujos nomes correspondem ao padrão são mostrados.

\dp [padrão]
Produz uma lista contendo todas as tabelas disponíveis junto com seus privilégios de acesso associados. Se padrão for especificado, somente as tabelas cujos nomes correspondem ao padrão são mostradas. Os comandos GRANT e REVOKE são utilizados para definir privilégios de acesso. Veja o comando GRANT para obter mais informações.

\dT [padrão]
Mostra todos os tipos de dado, ou somente aqueles que correspondem ao padrão. A forma \dT+ do comando mostra informações adicionais.

\du [padrão]
Mostra todos os usuários do banco de dados, ou somente aqueles que correspondem ao padrão.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\edit (ou \e) [nome_do_arquivo]
Se o nome_do_arquivo for especificado, o arquivo é editado; após o fim da execução do editor, o conteúdo do arquivo é copiado de volta para o buffer de comando. Se nenhum argumento for fornecido, o buffer de comando corrente é copiado para um arquivo temporário, que é editado de forma idêntica. O novo buffer de comando é então analisado novamente de acordo com as regras normais do psql, onde o todo buffer é tratado como sendo uma única linha (Portanto, não podem ser gerados scripts dessa forma. Use o comando \i para fazer isso). Isto significa também que, se o comando terminar por (ou contiver) um ponto-e-vírgula, será executado imediatamente. Se não contiver, apenas permanecerá aguardando no buffer de comando.
Dica: O psql procura nas variáveis de ambiente PSQL_EDITOR, EDITOR e VISUAL (nesta ordem) o editor a ser usado. Se nenhuma delas estiver definida, então /bin/vi é executado.

\echo texto [...]
Envia os argumentos para a saída padrão, separados por um espaço e seguido por um caractere de nova-linha, o que pode ser útil para intercalar informações na saída dos scripts. Por exemplo:
=> \echo `date`
Tue Oct 26 21:40:57 CEST 1999
Se o primeiro argumento for -n (não entre apóstrofos) o caractere de nova-linha final não é enviado. Dica: Se for usado o comando \o para redirecionar a saída do comando, talvez se deseje utilizar \qecho em vez deste comando.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\encoding [codificação]
Define a codificação do conjunto de caracteres do cliente. Sem argumento, este comando mostra a codificação corrente.

\f [cadeia_de_caracteres]
Define o separador de campos para a saída de comando não alinhada. O padrão é a barra vertical (|). Veja também em \pset uma forma genérica de definir opções de saída.

\g [{ nome_do_arquivo | |comando }]
Envia o buffer de entrada de comando corrente para o servidor e, opcionalmente, salva a saída em nome_do_arquivo, ou envia a saída para outro interpretador de comandos do Unix para executar o comando. Um \g puro e simples é virtualmente equivalente ao ponto-e-vírgula. Um \g com argumento é uma alternativa "de uma só vez" para o comando \o.

\help (ou \h) [comando]
Fornece ajuda de sintaxe para o comando SQL especificado. Se o comando não for especificado, então o psql mostra todos os comandos para os quais a ajuda de sintaxe está disponível. Se comando for um asterisco ("*"), então é mostrada a ajuda de sintaxe para todos os comandos SQL.
Nota: Para simplificar a digitação, os comandos compostos por várias palavras não necessitam estar entre apóstrofos. Portanto, pode ser digitado \help alter table.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\H
Ativa o formato HTML de saída da consulta. Se o formato HTML já estiver habilitado, retorna ao formato de texto alinhado padrão. Este comando existe por compatibilidade e conveniência, mas veja em \pset a definição de outras opções de saída.

\i nome_do_arquivo
Lê a entrada no arquivo nome_do_arquivo, e executa como se tivesse sido digitada pelo teclado.
Nota: Se for desejado ver as linhas na tela ao serem lidas, deve ser definida a variável ECHO como all.

\l (ou \list)
Mostra o nome, dono e codificação do conjunto de caracteres de todos oa bancos de dados do servidor. Deve ser adicionado + ao nome do comando para ver, também, todas as descrições dos bancos de dados.

\lo_export loid nome_do_arquivo
Lê no banco de dados o objeto grande com OID igual a loid, e escreve em nome_do_arquivo. Observe que isto é quase imperceptivelmente diferente da função do servidor lo_export, que atua com a permissão do usuário como o qual o servidor de banco de dados processa, e no sistema de arquivos do servidor.
Dica: Use \lo_list para descobrir os OIDs dos objetos grandes.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\lo_import nome_do_arquivo [comment]
Armazena o arquivo em um objeto grande do PostgreSQL. Opcionalmente, associa o comentário fornecido junto com o objeto. Exemplo:
foo=> \lo_import '/home/peter/pictures/photo.xcf' 'uma fotografia minha'
lo_import 152801
A resposta indica que o objeto grande recebeu o identificador de objeto 152801, que deve ser lembrado para acessar o objeto novamente. Por esta razão, recomenda-se associar sempre um comentário inteligível a cada objeto. Estes podem ser vistos utilizando o comando \lo_list. Observe que este comando é quase imperceptivelmente diferente da função lo_import do servidor, porque atua como o usuário local no sistema de arquivos local, em vez do usuário do servidor no sistema de arquivos do servidor.

\lo_list
Mostra a lista de todos os objetos grandes do PostgreSQL armazenados neste instante no banco de dados, junto com os comentários fornecidos para os mesmos.

\lo_unlink loid
Remove do banco de dados o objeto grande com o OID igual a loid.
Dica: Use \lo_list para descobrir os OIDs dos objetos grandes.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\o [{nome_do_arquivo | |comando}]
Salva os resultados dos próximos comandos no arquivo nome_do_arquivo, ou envia os próximos resultados para um outro interpretador de comandos do Unix para executar o comando. Se nenhum argumento for especificado, a saída do comando será redefinida para a saída padrão. Os "resultados dos comandos" incluem todas as tabelas, respostas dos comandos e notificações obtidas do servidor de banco de dados, assim como a saída dos vários comandos de contrabarra que consultam o banco de dados (como o \d), mas não as mensagens de erro.
Dica: Para intercalar saída de texto entre os resultados dos comandos deve ser utilizado \qecho.

\p
Envia o buffer de comando corrente para a saída padrão.

\pset parâmetro [value]
Este comando define opções que afetam a saída das tabelas de resultado das consultas. O parâmetro indica qual opção será definida. A semântica do valor depende do parâmetro. As opções ajustáveis de exibição são: format, border, etc...

- Veja nos próximos slides

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

format
Define o formato de saída como unaligned (não alinhado), aligned (alinhado), html ou latex. Formas abreviadas únicas são permitidas (O que equivale a dizer que uma letra basta). O modo "Unaligned" escreve todos as colunas em uma linha, separadas pelo separador de campos ativo no momento. Pretende-se com isso poder criar uma saída que sirva de entrada para outro programa (separada por tabulação, por vírgula, etc.). O modo "Aligned" é a saída de texto padrão, inteligível e agradavelmente formatada. Os modos "HTML" e "LaTeX" produzem tabelas feitas para serem incluídas em documentos usando a linguagem de marcação correspondente. Não são documentos completos! (Isto não é tão problemático no HTML, mas no LaTeX deve haver um invólucro completo do documento).

border
O segundo argumento deve ser um número. Em geral, quanto maior o número mais bordas e linhas a tabela terá, mas isto depende do formato. No modo HTML será traduzido diretamente para o atributo border=..., nos outros modos somente os valores 0 (sem borda), 1 (linhas divisórias internas) e 2 (moldura da tabela) fazem sentido.

expanded (ou x)
Alterna entre os formatos regular e expandido. Quando o formato expandido está ativo, todas as saídas possuem duas colunas, com o nome da coluna à esquerda e o dado à direita. Este modo é útil quando os dados não cabem na tela no modo normal "horizontal". O modo expandido é suportado por todos os quatro modos de saída.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

null
O segundo argumento é a cadeia de caracteres a ser mostrada sempre que o campo for nulo. O padrão é não mostrar nada, que pode ser facilmente confundido com, por exemplo, uma cadeia de caracteres vazia. Portanto, pode ser preferido escrever \pset null '(nulo)'.

fieldsep
Especifica o separador de campos a ser utilizado no modo de saída não alinhado. Desta forma pode ser criada, por exemplo, uma saída separada por tabulação, por vírgula ou por um outro caractere, conforme o programa preferir. Para definir o caractere de tabulação como separador de campo deve ser usado \pset fieldsep '\t'. O separador de campos padrão é o '|' (a barra vertical).

footer
Alterna a exibição do rodapé padrão (x linhas).

recordsep
Especifica o separador de registro (linha) a ser usado no modo de saída não alinhado. O padrão é o caractere de nova-linha.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

tuples_only (ou t)
Alterna entre mostrar somente as tuplas e mostrar tudo. O modo mostrar tudo pode mostrar informações adicionais como os cabeçalhos das colunas, títulos e vários rodapés. No modo tuplas-apenas somente os dados da tabela são mostrados.

title [texto]
Define o título para as próximas tabelas mostradas. Pode ser usado para colocar textos descritivos na saída. Se nenhum argumento for fornecido, o título é removido.

tableattr (ou T) [texto]
Permite especificar qualquer atributo a ser colocado dentro da marca table do HTML. Estes atributos podem ser, por exemplo, cellpadding ou bgcolor. Observe que, provavelmente, não será desejado especificar border aqui, porque isto já é tratado pelo \pset border.

pager
Controla o uso do paginador para comandos e para a saída da ajuda do psql. Se a variável de ambiente PAGER estiver definida, a saída é enviada para o programa especificado, senão o padrão dependente da plataforma é utilizado (como o more). Quando o paginador está inativo, a paginação não é feita. Quando o paginador está ativo, a paginação é feita somente quando for apropriado, ou seja, a saída é para um terminal e não cabe na tela. O (psql não realiza um trabalho perfeito ao avaliar quando o paginador deve ser utilizado). \pset pager ativa e desativa o paginador. O paginador também pode ser definido como always, o que faz o paginador ser sempre utilizado.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

Existem vários comandos abreviados para o \pset. Veja \a, \C, \H, \t, \T e \x.
Nota: É errado chamar o \pset sem argumentos. No futuro, esta chamada deverá mostrar o status corrente de todas as opções de exibição.

\q
Sair da aplicação psql.

\qecho texto [...]
Este comando é idêntico a \echo, exceto que toda a saída é enviada para o canal de saída de comando, definido por \o.

\r
Restaura (limpa) o buffer de comando.

\s [nome_do_arquivo]
Mostra ou salva o histórico da linha de comando em nome_do_arquivo. Se nome_do_arquivo for omitido, o histórico é enviado para a saída padrão. Esta opção somente estará disponível se o psql estiver configurado para usar a biblioteca de histórico GNU.
Nota: Na versão corrente não é mais necessário salvar o histórico dos comandos, porque isso é feito automaticamente quando o programa termina. O histórico também é carregado, automaticamente, toda vez que o psql inicia.

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\set [nome [value [...]]]
Define a variável interna nome com o valor ou, se mais de um valor for fornecido, com a concatenação de todos os valores. Se o segundo argumento não for fornecido, a variável é definida sem valor. Para remover a definição da variável deve ser usado o comando \unset. Nomes de variáveis válidos podem conter letras, dígitos e sublinhados (_). Veja a seção Variáveis abaixo para obter detalhes. Embora possa ser definida qualquer variável como qualquer coisa desejada, o psql trata várias variáveis como sendo especiais. Elas estão documentadas na seção sobre variáveis.
Nota: Este comando é totalmente distinto do comando SQL SET.

\t
Alterna a exibição do cabeçalho contendo o nome das colunas e do rodapé contendo o número de linhas. Este comando equivale a \pset tuples_only, sendo fornecido por conveniência.

\T opções_de_tabela
Permite especificar atributos a serem colocados na marca table no modo de saída tabular HTML. Este comando equivale a \pset tableattr opções_de_tabela.

\timing
Alterna a exibição de quanto tempo cada comando SQL leva, em milissegundos.

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Terminal Interativo – PSQL – Meta dados

\w {nome_do_arquivo | |command}
Escreve o buffer de comando corrente no arquivo nome_do_arquivo, ou envia para o comando Unix comando através de um pipe.

\x
Alterna o modo de formatação de tabela estendido. Sendo assim, equivale a \pset expanded.

\z [padrão]
Produz uma lista contendo todas as tabelas disponíveis, junto com seus privilégios de acesso. Se padrão for especificado, somente as tabelas cujos nomes correspondem ao padrão são listadas. Os comandos GRANT e REVOKE são utilizados para definir os privilégios de acesso. Veja o comando GRANT para obter mais informações. Este comando é um outro nome para o comando \dp ("display privileges").

\! [comando]
Abre um outro interpretador de comandos do Unix, ou executa o comando Unix comando. Os argumentos não são mais interpretados, sendo enviados para o interpretador de comandos como estão.

\?
Mostra informação de ajuda para os comandos de contrabarra ("\").

Aula 03

PostgreSQL

Utilizando o PSQL

A última linha exibida pelo psql é o prompt, indicando que o psql está aguardando você, e que você pode digitar comandos SQL dentro do espaço de trabalho mantido pelo psql. Tente usar estes comando:

```
template1=> SELECT version();
                          version
------------------------------------------------------------------
PostgreSQL 8.0.0beta3 on i686-pc-mingwn32, compiled by GCC 3.2.3
(1 row)

template1=> SELECT current_date;
   date
------------
 2004-10-21
(1 row)

template1=> SELECT 2 + 2;
 ?column?
----------
        4
(1 row)
```

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Manipulando a base de dados passo a passo.

1) Criar uma tabela chamada filmes

```
template1=> create table filmes (
                        nome varchar(40),
                        ano smallint,
                       diretor varchar(30));
CREATE
template1=>
```

No exemplo foi criada uma tabela de nome filmes com os campos:

nome, tipo varchar com tamanho máximo de 40.
ano, tipo smallint.
diretor, tipo varchar com tamanho máximo de 30 caracteres.

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

2) Inserir dados dentro desta tabela

```
template1=> insert into filmes values
template1-> ('Morango e Chocolate',1993,'Tomas Gutierrez');
INSERT 18433 1

template1=> insert into filmes values
template1-> ('Guerra nas Estrelas I',1980,'Spilberg');
INSERT 18434 1

template1=> insert into filmes values
template1-> ('Guerra nas Estrelas II',1980,'Spilberg');
INSERT 18435 1
```

A sintaxe do comando é:
INSERT INTO table VALUES (val1,val2,...,valN);
Note que as strings devem ser delimitadas por aspas simples.

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Para acessar os dados da tabela precisamos fazer uma query.

```
template1=> select * from filmes;
nome                          | ano |diretor
------------------------------+-----+----------------------
Morango e Chocolate           |1993|Tomas Gutierrez
Guerra nas Estrelas I         |1990|Spilberg
Guerra nas Estrelas II        |1990|Spilberg
(3 rows)
```

Usamos o comando Select para fazer as consultas.

SELECT campo1,campo2,...,campoN FROM table;

Veja outros exemplo:

```
template1=> select nome,ano from filmes;
nome                          | ano
------------------------------+----
Morango e Chocolate           |1993
Guerra nas Estrelas I         |1990
Guerra nas Estrelas II        |1990
(3 rows)
```


Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

```
template1=> select * from filmes where diretor='Spilberg';
nome                          | ano |diretor
------------------------------+-----+----------
Guerra nas Estrelas I         |1990|Spilberg
Guerra nas Estrelas II        |1990|Spilberg
(2 rows)
```

No segundo exemplo vemos a cláusula where que serve para selecionar os registros que queremos ver. A sintaxe é
WHERE campo1 operador valor1,..., campoN operador valorN

Outra claúsula interessante é a ORDER BY que permite mostrar uma tabela ordenada por um campo. Veja o exemplo:

```
template1=> select * from filmes order by nome;
nome                          | ano |diretor
------------------------------+-----+----------------------
Guerra nas Estrelas I         |1990|Spilberg
Guerra nas Estrelas II        |1990|Spilberg
Morango e Chocolate           |1993|Tomas Guitierrez Alea
(3 rows)
```

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Cometemos um erro nos dados que entramos, ``Guerra nas Estrelas II'' não foi feito no mesmo ano que o primeiro. Para corrigir isso vamos executar o seguinte comando

```
template1=> update filmes set ano=1991
where nome='Guerra nas Estrelas II';
UPDATE 1
```

A sintaxe do UPDATE é
UPDATE table name SET campo1=valor1,..., campoN=valorN

A claúsula WHERE do UPDATE é igual à do SELECT. Ela é muito importante pois diz para o postgres quais registros devem ser modificados, se você não coloca-la ele modifica todos os registros. Veja abaixo como a tabela ficou depois do UPDATE:

```
template1=> select * from filmes;
nome                          | ano  |diretor
------------------------------+------+---------------------
Morango e Chocolate           |1993 |Tomas Gutierrez
Guerra nas Estrelas I         |1990 |Spilberg
Guerra nas Estrelas II        |1991 |Spilberg
(3 rows)
```


Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Muito parecido com o comando UPDATE é o comando DELETE. Nele também claúsula WHERE é muito importante, se você esquece-la vai apagar todo o seu banco de dados, por isso digite-o com cuidado.

```
template1=> delete from filmes where diretor='Tomas Gutierrez';
DELETE 1
```

A sintaxe do DELETE é muito simples, novamente a clausula WHERE é igual ao do SELECT. Note que eu não a coloquei na sintaxe do comando. É porque ela não é obrigatória, apesar de raramente o DELETE ser usado em ela.

DELETE FROM table name

Esses foram os comandos básicos da linguagem SQL. Agora vamos ver alguns tópicos mais avançados. O primeiro deles serão os índices. Eles são muito importantes pois tornam as querys com clausúla WHERE e ORDER BY muito mais rápidas. Você pode indexar qualquer campo de uma tabela, só que quanto mais campos você indexar mais lenta ficam as operações INSERT e UPDATE. Vamos então indexar o campo nome.

```
template1=> create unique index filmes_nome on filmes (nome);
CREATE
```

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

A sintaxe de CREATE INDEX é

CREATE INDEX index name ON table name (campo)

No nosso exemplo usamos a claúsula opcional unique que faz com que o postgres não aceite informações repetidas no campo indexado. Ou seja na nossa tabela não pode haver dois filmes com o mesmo nome. Claro que isso não faria sentido no campo ano.

Vamos supor que você comprou todos os filmes do Spilberg e quer cadastra-los no seu banco de dados. Em cada um dos registros você vai ter digitar o nome do diretor, se mais e uma pessoa for digitar os registros uma pode digitar "Steven Spilberg", outro digita somente "Spilberg", da para imaginar o problema que você teria para fazer uma query que mostrasse todos os filmes do Spilberg. Uma solução para esse problema é você criar duas tabelas, uma para os filmes e outra para os diretores, e relacionar as duas por um campo, de preferência numérico. Apague a tabela que você criou com comando:

```
template1=> drop table filmes;
DROP
```

E crie uma nova tabela filmes

```
template1=> create table filmes (nome varchar(40), ano int, diretor int);
CREATE
```


Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Note que o campo diretor agora é um numero inteiro. Como vamos ter uma tabela só para diretores, podemos por informações ligadas ao diretor nessa tabela, como por exemplo, a nacionalidade dele. Veja que podemos por quantas informações quisermos na tabela de diretores que ela não vai se repetir em cada registro da tabela filmes, assim economizamos espaço em disco e evitamos informações redundantes.

```
template1=> create table diretores (id int, nome char(30),
nacionalidade char(10), unique(id));
NOTICE:  CREATE TABLE/UNIQUE will create implicit index
diretores_id_key for table diretores
CREATE
```

Note que a sequencia unique(id) já indexa o campo id da tabela diretores. Esse campo id é muito importante, pois é com ele que nós vamos relacionar o campo diretores na tabela filmes.

Antes de inserir os filmes vamos inserir os diretores

```
template1=> insert into diretores values (0,'Steven Spilberg','Americano');
INSERT 18527 1
template1=> insert into diretores values (1,'Ewa Karpoff','Polones');
INSERT 18528 1
```

PROF. CLÁUDIO FARIAS ROSSONI

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

Agora insira os filmes

```
template1=> insert into filmes values ('A volta dos que nao foram',1990,1);
INSERT 18529 1
template1=> insert into filmes values ('A volta dos que nao foram II',1991,1);
INSERT 18530 1
template1=> insert into filmes values ('De volta pro futuro',1985,0);
INSERT 18531 1
```

Veja que no campo diretor nós colocamos o número id dele que aparece na tabela de diretores.

Tente agora mostrar os dados da tabela filmes

```
template1=> select * from filmes;
A volta dos que nao foram        |1990|     1
A volta dos que nao foram II     |1991|     1
De volta pro futuro              |1985|     0
```

Usando duas tabelas relacionadas nós eliminamos o problema de dados redundantes, mas criamos outro. Agora em vez de aparecer o nome dos diretores aparece um monte de números, algo difícil para seres humanos lidarem. Mas não precisa se preocupar com um query um pouco mais complexa podemos resolver esse problema.

Aula 03

PostgreSQL

Manipulando Base de Dados – PSQL

```
template1=> select filmes.nome,filmes.ano,diretores.nome from
filmes,diretores where filmes.diretor=diretores.id;
De volta pro futuro              |1985|Steven Spilberg
A volta dos que nao foram        |1990|Ewa Karpoff
A volta dos que nao foram II     |1991|Ewa Karpoff
```

Veja que a clausula from tem duas tabelas, isso é necessário porque a informação agora está espalhada por mais de uma tabela. A capacidade de juntar várias tabelas em uma única query é a capacidade mais importante de linguagem SQL. Veja também que a agora temos dois campos nome, um na tabela filmes e outro na tabela diretores, para diferencia-los especificamos o nome da tabela antes do nome do campo, separando os dois por um ponto. Outra coisa importante é a clausula WHERE que diz para o postgres que o campo diretor da tabela filmes se relaciona com campo id da tabela diretores.

PROF. CLÁUDIO FARIAS ROSSONI


PostgreSQL
Término da Aula
Aula 03
PROF. CLÁUDIO FARIAS ROSSONI