Sistemas de Bancos de Dados
Instalação do Postgres, To join or not to Join e funções

O objetivo desta aula é instalarmos o módulo servidor do PostgreSQL e 'subir' o banco IB em suas máquinas virtuais de modo a ficarmos independentes do servidor remoto. A maior parte do nosso curso visa a aplicação da SQL embutida em uma linguagem de programação, neste caso, a linguagem java. Dessa forma subentende-se que o uso da SQL não é mais uma barreira para o aluno transpor, mas a sua utilização conjugada a uma linguagem de programação. Para início dos trabalhos, será necessário instalar e configurar o postgres.

1 Caso você encontre problemas em instalar algum programa então você deve instalar um buscador de programas possíveis de serem instalados na versão do seu OS. Para tanto instale o programa:

1.1 sudo apt install aptitude

1.2 Para buscar por um programa digite, por exemplo: aptitude search postgres

2 Iniciamos com a instalação do postgres. Para tanto digite no terminal o comando:

2.1 sudo apt install postgresql-12

Dependendo da versão do Ubuntu a versão do postgres será diferente. Para a versão 20.04 do Ubuntu é a versão 12.

2.2 sudo apt install pgadmin3

3 Agora devemos proceder com a criação de uma senha para o usuário *postgres* que foi automaticamente instalado no momento que o SGBD foi instalado na máquina. Para tanto, devemos usar o nosso usuário *sudoer* para modificar a senha do usuário *postgres* para um valor conhecido. Usaremos como senha o mesmo nome do usuário. Execute o comando que permitira a digitação de uma nova senha para o usuário postgres

3.1 sudo passwd postgres

4 Acabamos de criar a senha do usuário postgres no OS, mas ainda precisamos alterar a senha do usuário *postgres* que existe no banco de dados (são usuários diferentes, apesar de terem o mesmo nome). Para modificar a senha do usuário postgres no sgbd, é necessário primeiro fazer o login com o usuário postgres no OS e a partir dele acessar o banco de dados. O segredo aqui é que apesar de não termos modificado a senha do usuário de banco de dados de nome postgres, a este usuário do OS não será solicitada senha para acesso ao banco de dados uma vez que este usuário do OS é o proprietário do banco de dados. Para tanto, faça o login com o usuário postgres do OS utilizando o comando su. O comando su permite executar o login de um outro usuário (postgres) mesmo estando no login de outro usuário (seu usuário):

4.1 su postgres (digite a senha criada no passo anterior)

Para acionar a interface de linha de comando para comando SQL do SGBD digite:

4.2 psql

5 Acabamos de logar no banco de dados principal do postgres, um banco que é criado no momento de instalação do programa. Para logar neste banco não é necessário senha visto que o usuário postgres é o proprietário do banco. Aqui ficam, por exemplo, os dados de usuários criados para os bancos de dados que forem criados nesta máquina. Devemos alterar a senha do usuário postgres para o mesmo valor utilizando o seguinte comando:

```
ALTER USER postgres WITH ENCRYPTED PASSWORD 'postgres';
```

6 Alterada a senha do usuário postgres, agora podemos utilizá-la para nos conectarmos a este servidor de banco de dados por meio do programa pgadmin3. Utilizando os menus do OS inicie o programa pgadmin3 para criarmos um banco de dados vazio que será utilizado para hospedar o nosso banco de dados de uma instituição bancária (IB). Outra possibilidade é digitar no terminal de comandos:

createdb -h localhost -p 5432 -U postgres IB

Uma vez criado o banco de dados vazio de nome IB, precisamos realizar a carga de dados no banco. Para tanto devemos:

6.1 Baixar o arquivo com o banco de dados do link: https://www.dropbox.com/s/a04xkmtvmm9wvo0/IB.dump

6.2 Realizar a carga do IB.dump no banco de dados IB. Lembre-se que a última vez que a sua janela de comandos foi manuseada, de acordo com este tutorial, você estava logado como o usuário postgres. Para evitarmos problemas de permissões de acessos a arquivos entre o usuário postgres e o login com o qual você acionou o OS, devemos fechar a sessão do usuário postgres com os comandos:

6.2.1 \q → para sair do psql;

6.2.2 exit → para sair do usuário postgres

6.3 Agora posicione-se no diretório no qual o download do arquivo IB.dump foi efetuado, por exemplo, com 'cd /home/login_usuário/Downloads' e digite o comando:

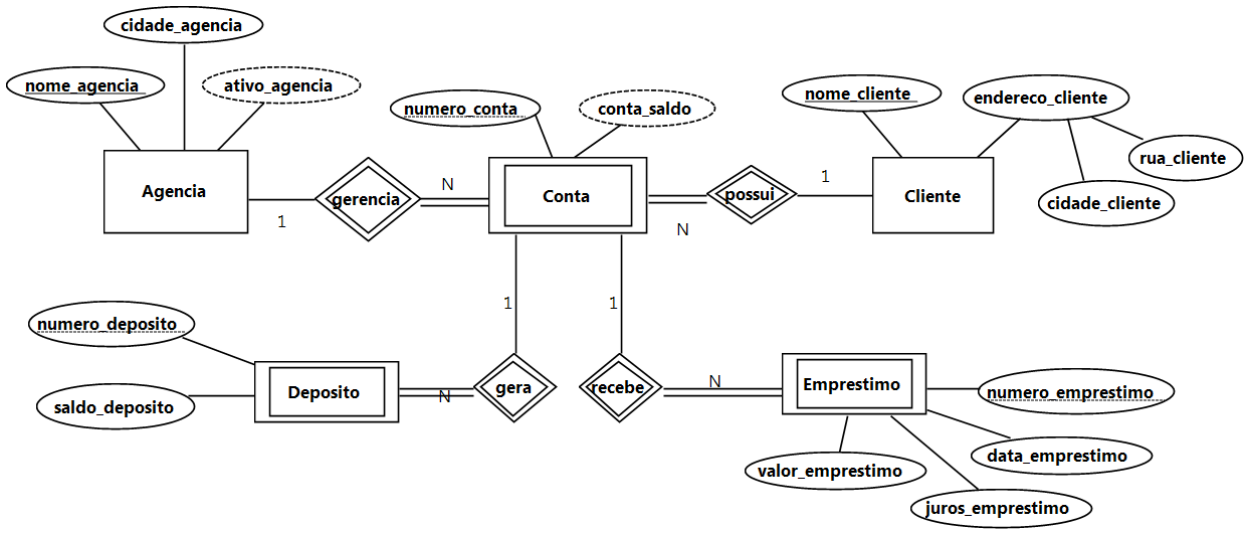
psql -h localhost -p 5432 -U postgres -f IB.dump IB

Esse comando vai criar tabelas e popular de dados essas tabelas. Assim teremos um BD pronto para execução de diversas consultas em SQL que estão nos tópicos de 10 a 20 dessa aula.

6.4 Agora a criação de uma conexão ao banco IB pelo pgadmin3 exibirá o BD de uma instituição bancária hipotética. Use os mesmos parâmetros dos comandos psql para criar esta conexão via o pgadmin3 ao banco IB.

7 A figura abaixo exibe o esquema do nosso banco de dados IB. Ele será útil para nos lembrarmos da estrutura para fins de realização de diversas consultas daqui em diante.

Sistemas de Bancos de Dados
Instalação do Postgres, To join or not to Join e funções



8 A primeira tarefa sobre o banco de dados IB será uma comparação entre os tempos de execução de tabelas que realizam junções por meio da teoria de conjuntos e por meio da cláusula JOIN da SQL. Para este propósito, vamos aumentar de forma significativa a quantidade de linhas armazenadas em uma das tabelas.

9 Acrescente, por exemplo, 100.000 depósitos de R$ 1,00 (Um Real) na conta do cliente 'Germano Luiz de Paula', na agência 'Pampulha', na conta 93134. Você deve executar o código abaixo em uma janela de comandos do PostgreSQL, quando o banco de dados selecionado para consultas for o nosso banco IB.

```
DO
$do$
BEGIN
FOR num IN 1..100000 LOOP
       insert into deposito values (num, 93134, 'Pampulha', 'Germano Luiz de Paula',
1.00);
END LOOP;
END
$do$
```

10 É possível que o trecho de código acima seja uma novidade para alguns; podemos executar comandos procedurais de programação PL/SQL caso eles estejam dentro de uma função ou dentro de uma cláusula 'DO'. O código possui um contador FOR que vai de 1 até 1000 e cada vez que este comando FOR passa por uma destas contagens executa o comando *insert*. O comando *insert* possui como número de depósito (chave primária de depósito) o número contado pelo FOR. Nesse caso vai funcionar porque, no nosso banco IB, o menor número de depósito era (até antes da execução do código acima) um número maior que 100 mil, um número gerado de forma aleatória quando eu criei o banco IB.

11 Agora que temos o banco IB com uma tabela de tamanho significativamente maior, podemos testar a hipótese levantada no texto introdutório: o uso da cláusula JOIN é cômodo, mas custoso.

Lembra-se de nossa velha consulta:

Sistemas de Bancos de Dados
Instalação do Postgres, To join or not to Join e funções

11.1 *Selecione os nomes dos clientes e seus respectivos números de conta e nome de agência que fizeram depósitos e empréstimos ao mesmo tempo*.

Este código em SQL resolve esta consulta sem fazer uso da cláusula JOIN:

```
select nome_cliente, numero_conta, nome_agencia from emprestimo
intersect
select nome_cliente, numero_conta, nome_agencia from deposito
```

Agora vem a sua tarefa:

11.2 Construa a consulta equivalente a este exemplo utilizando a cláusula JOIN.

11.3 Construa a consulta equivalente a este exemplo utilizando a SELECT DISTINCT e sem o JOIN.

12 Construa uma tabela em uma planilha (Programa *Calc* do *Libre Office*) com três colunas: *intersect*, *distinct* e *join*. Execute as três consultas pelo menos 30 vezes e registre na planilha o tempo de execução em milissegundos para a conclusão de cada uma das três versões da consulta. O tempo de execução de cada consulta é exibido no canto inferior direito da tela que executou uma consulta. Ao final, tire a média de cada coluna e conclua qual versão da consulta foi mais rápida.

13 Uma das principais vantagens obtidas pela utilização da SQL está no uso de uma linguagem não procedural, ou seja, é necessário dizer apenas “o que” é desejado que seja retornado em uma consulta sem se preocupar com os detalhes de “como” a consulta será executada. Entretanto, em determinadas situações nos encontramos enfrentando problemas nos quais uma simples consulta feita na SQL não nos atende. Vejamos alguns exemplos:

Cenário 1) Você precisa executar uma consulta em SQL para retornar os dados de clientes e para cada cliente será criada uma única linha em um relatório. Até aqui podemos resolver facilmente com uma simples consulta em SQL. Entretanto, foi solicitado que os números de todas as contas de um cliente, de cada agência, sejam exibidos nesta mesma linha destinada aos dados do cliente. Neste caso, haverá uma coluna no relatório que vai juntar o nome da agência ao número da conta e este par de dados será exibido como um só dado (exemplo: “Central-12345”). Quantas contas e agências tiver um cliente devem ser exibidas nesta coluna separado por vírgula (exemplo: “Central-12345”, “Pampulha-67890”, …). E agora? Como você vai retornar todas as contas do cliente em uma única consulta, de modo que essa lista passe a constituir uma das colunas de dados a serem retornadas?

Cenário 2) Novamente, você precisa executar uma consulta em SQL para retornar dados de clientes. Os dados armazenados te permitem inferir o quão interessantes são estes clientes para receberem um novo tipo de cartão, com melhores taxas e mais crédito. Para este propósito foram criadas três faixas de clientes: A , B e C cujas as somas dos valores depositados ultrapassem seis mil, quatro mil e um mil Reais, respectivamente. O foco principal são os clientes classes A, mas existe a possibilidade de que uma parte dos novos clientes deste cartão seja oriunda de clientes do tipo B. Desta forma solicita-se que no relatório final conste apenas o nome do cliente (e outros dados de contato, por exemplo) acompanhado da letra que denomina a faixa de classificação do cliente, as somas das quantias depositadas não devem ser exibidas no relatório. Novamente eu te pergunto: como você vai codificar em letras as faixas de depósitos dos clientes em uma única linha de consulta SQL?

Sistemas de Bancos de Dados
Instalação do Postgres, To join or not to Join e funções

Como o assunto desta aula trata de funções e procedimentos em SQL, então a resposta aos questionamentos feitos acima ficam fáceis, concorda?

As perguntas que te fiz nestes cenários foram um pouco capciosas porque ainda que sua consulta conste de apenas uma linha de código SQL, ela fará a chamada de uma função. Nesse caso a função terá um conjunto extra de linhas de código SQL com algum código procedural na forma de expressões condicionais, laços (*loops*) e até possivelmente uma ou mais consultas SQL embutidas. Veja estas alternativas para resolver os problemas dos cenários:

Possível solução do Cenário 1) Implementar uma função que, utilizando os dados identificadores de um cliente, pesquisa na tabela de contas especificamente por registros de um único cliente: o cliente cuja linha esteja sendo escrita como saída do relatório final. Utilizando um *loop* para iterar entre todos os resultados da consulta realizada somente aos dados deste único cliente, todos os números de conta e suas respectivas agências seriam retornados, um a um, para uma variável de texto que concatenasse os dados na forma "*Nome da Agência-Número da Conta*". Em seguida esta variável de texto deveria também ser concatenada a uma outra variável de texto com o propósito de criar uma lista de contas de cada cliente. Quando o *loop* terminasse a sua execução, por não haver mais registros a serem retornados para um determinado cliente, então todas as contas deste cliente estariam presentes na variável lista e esta seria retornada pela função. Ao receber o valor de retorna da função a nossa consulta inicial utilizará este valor para ocupar a coluna de contas para o cliente da vez, ou seja, o cliente cujos dados estão sendo escritos no relatório final.

Possível solução do Cenário 2) Implementar uma função que, utilizando os dados identificadores de um cliente, pesquisa na tabela de depósitos especificamente por registros de um único cliente: o cliente cuja linha esteja sendo escrita como saída do relatório final. Todos os depósitos deste cliente único seriam somados em uma variável. Em seguida esta variável seria submetida a estruturas de decisão do tipo *if-then-else* para determinar se a letra que deve ser retornada pela função é A, B ou C. A letra retornada pela função ocuparia então a coluna de classificação na linha do cliente especificamente consultado.

13.1 Implemente primeiro (por ser mais simples) uma função em PL/pgSQL para retornar o relatório descrito no cenário 2 a partir desta consulta SQL;

```
select faixa_cliente(nome_cliente), nome_cliente from cliente;
```

13.2 Implemente uma função em PL/pgSQL para retornar o relatório descrito no cenário

```
select nome_cliente, contas_cliente(nome_cliente), cidade_cliente from cliente;
```

Lembre-se de consultar o manual de comandos da PL/pgSQL para ver a sintaxe de cada comando. Fique à vontade para partir dos códigos de exemplos de outras aulas para criar as suas funções.

Entregue um relatório em PDF das três tarefas: instalação do postgres, To join or not to join e funções. O código gerado deve ser entregue em separado do relatório PDFpara que o professor possa compilar e rodar em outra máquina.