Design e Desenvolvimento de Bancos de Dados
— *PSet 1* —

Prof. Abrantes Araújo Silva Filho

Data de Entrega: 26/04/2022

LEIA COM ATENÇÃO!

Um *Problem Set* (PSet) é um conjunto de problemas e tarefas difíceis (alguns extremamente difíceis) que o forçarão a estudar e realmente compreender a matéria[1]. É **essencial** que você inicie o PSet o mais rápido possível... deixar para começar um PSet faltando um ou dois dias da data de entrega significa que você não conseguirá terminar. Também é **fundamental** que você, em caso de dúvidas ou dificuldades, participe das monitorias e discuta suas dúvidas com o monitor.

Sumário

[1] Para maiores informações sobre os PSets, leia a Seção 5.3 do *Syllabus* da disciplina.

1 Instruções

Este PSet é uma das atividades **pontuadas** que, conforme detalhado no *Syllabus* (Seções 5 e 6), terão peso de 45% na nota final da disciplina. Por favor siga todas as instruções a seguir para a realização e entrega deste PSet.

Este PSet tem quatro grandes objetivos:

1. Fazer com que você aprenda sobre sistemas de controles de versões, em especial o Git (https://git-scm.com/) e o serviço web GitHub (https://github.com) e, ainda, aprender sobre Markdown (em especial a versão GitHub Flavored Markdown);
2. Fazer com que você aprenda a implementar projetos lógicos em diversos Sistemas de Gerenciamento de Bancos de Dados (SGBD), utilizando os quatro SGBD relacionais mais comuns e importantes hoje em dia: (PostgreSQL, Oracle, MariaBD/MySQL e MS SQL Server);
3. Fazer com que você reflita sobre diversos problemas que podem ocorrer se o projeto lógico está mal preparado;
4. Fazer com que você aprenda a Structured Query Language (SQL) em nível básico e intermediário.

O PSet pode ser feito em grupos de até três alunos desde que: a) **cada aluno escreva individualmente sua própria resposta**; b) **você esteve envolvido em todos os aspectos do PSet**; c) **cada aluno escreva e comente seu próprio conjunto de código**; e d) **você identifique, nas suas respostas e nos seus códigos, quem trabalhou com você**.

Atenção para as regras de trabalho colaborativo!

Por favor, **LEIA ATENTAMENTE** as regras sobre Integridade Acadêmica do *Syllabus* (Seção 7), em especial a "**Política sobre trabalho colaborativo**" (Seção 7.1), para você saber **como trabalhar em grupo** nesta disciplina. Atividades que quebrarem essas regras terão a nota zerada e os alunos encaminhados à coordenação da UVV para aplicação das penalidades previstas (exceto quando o aluno utilizar a "**Cláusula de Arrependimento**", Seçao 7.2 do *Syllabus*).

As atividades deste PSet devem ser realizadas, preferencialmente, na **Máquina Virtual** da disciplina, que já tem o ambiente todo configurado, os softwares instalados e está pronta para uso. A máquina virtual pode ser baixada no site do Computação Raiz (https://www.computacaoraiz.com.br). Caso você opte por não utilizar a máquina virtual, terá de instalar todos os SGBD em seu próprio computador e configurar o ambiente por conta própria (é difícil, mas não impossível).

Antes de começar, leia integralmente o PSet para ter uma noção geral do que será exigido, para estimar o grau de dificuldade que você terá, e para estimar quantas

horas por dia você precisará se dedicar ao PSet. Em caso de dúvidas, entre em contato com o professor ou com os monitores o mais rápido possível.

Lembre-se: deixar para começar um PSet faltando apenas um ou dois dias da data de entrega significa que você não conseguirá terminar! Comece cedo, trabalhe um pouco todo dia, mantenha-se atento à data de entrega e tire suas dúvidas com os monitores e/ou professor. Não ultrapasse a data de entrega!

1.1 Como entregar este PSet?

A entrega do resultado deste PSet será feita de duas maneiras simultâneas:

- **PAPEL ALMAÇO**: algumas questões são discursivas e você as responderá por escrito, de forma **MANUSCRITA**, utilizando-se apenas e unicamente folhas de papel almaço pautadas no formato A4. Considere o seguinte:
  - Atividades entregues em outro tipo de papel não serão corrigidas e o aluno ficará com 0 (zero) nestas questões do PSet;
  - Escreva um cabeçalho contendo: "PSet 1", seu nome completo e sua turma;
  - As respostas podem ser escritas com caneta ou lápis. Os lápis devem ser escuros (ex.: 2B, 4B, 6B). Você pode utilizar lápis de cor para desenhar e/ou ilustrar suas respostas, se for o caso;
  - As respostas devem estar em **LETRA LEGÍVEL** e o texto deve seguir a **NORMA CULTA** da língua portuguesa;
  - Caso o professor considere, ao seu único e exclusivo critério, que o texto está ilegível, a questão ou texto considerado receberá nota 0 (zero) e não será corrigida;
  - Erros gramaticais graves receberão penalidades de 0,05 pontos por erro;
  - A apresentação do PSet é importante! Respostas mal organizadas, bagunçadas, rasuradas, sujas, manchadas etc. sofrerão penalidades de 0,5 pontos.
- **REPOSITÓRIOS GIT**: em algumas questões você precisará escrever código SQL, scripts executáveis diretamente pelos SGBD, documentação em formato Markdown e outras coisas semelhantes. Essas entregas serão feitas através de um repositório Git que você deve criar no GitHub[2].
- Leia atentamente cada questão do PSet pois elas indicam exatamente como devem ser feitas as entregas.

[2]Se você não sabe user o Git e/ou o GitHub, não se preocupe: as primeiras questões do PSet existem justamente para que você aprenda a utilizar essas ferramentas.

2 Git e GitHub

2.1 Git

O **Git** (https://git-scm.com) é um **Sistema de Controle de Versões**, um dos mais utilizados e importantes hoje em dia. Praticamente nenhum desenvolvedor trabalha sem utilizar um bom VCS (do inglês, *Version Control System*).

Um VCS é um software que nos permite controlar todas as alterações em um código fonte (ou demais documentos), nos permite criar ramos (*branches*) separados de codificação para a correção de bugs ou testes de novas funcionalidades, nos permite saber quem, quando e porquê uma determinada alteração no código foi feita e, principalmente, mantém um histórico recuperável de todo o código fonte, desde que a primeira linha de código foi escrita.

Sua primeira atividade neste PSet é aprender mais sobre os VCS, em geral, e sobre o Git, em especial. Você deve fazer o seguinte:

- Leia sobre os VCS na Wikipedia[3] ou procure vídeos no YouTube;
- Procure entender o que são, o que fazem e qual a importância dos diversos VCS;
- Entenda o que são VCS centralizados e distribuídos, comerciais e *open-source*;
- Visite o site do Git[4] e leia diversas páginas do site, procurando entender o Git e suas vantagens;
- Faça o download do livro "Pro Git", de Scott Chacon & Ben Straub, que está disponível integralmente no site do Git. **Leia os capítulos 1 e 2!**, isso é fundamental: para aprender a usar o Git com eficiência, você deve ler esses dois capítulos iniciais do livro![5] No site do Git existe uma versão online traduzida para Português caso você não leia em inglês[6].
- Procure vídeos ou tutoriais sobre o Git na internet, para aprender.

2.2 GitHub

Agora que você já sabe e domina o básico do Git, está na hora de aprender a utilizar um dos maiores provedores de servidores Git na internet, o **GitHub** (https://github.com).

Sua próxima atividade é aprender o que é o GitHub: O que ele faz? Para que serve? Como utilizar? GitHub é a mesma coisa que Git? É gratuito ou pago? Faça o seguinte:

[3] https://en.wikipedia.org/wiki/Version_control
[4] https://git-scm.com
[5] O resto você deve ler à medida que necessitar de funcionalidades mais avançadas em seu trabalho.
[6] E se esse for o seu caso, já passou da hora de estudar inglês: **tudo na computação depende de inglês**.

- Procure vídeos e tutoriais na internet que ensinem como utilizar o GitHub.
- Visite o GitHub do professor (https://github.com/abrantesasf) e veja alguns dos repositórios públicos que estão lá desde 2012. Note que vários repositórios foram arquivados no Cofre Ártico de Código do GitHub!
- Conheça o Programa de Arquivamento do GitHub (https://archiveprogram.github.com/) e assista (somente em inglês) ao vídeo do Cofre Ártico de Código (https://archiveprogram.github.com/arctic-vault/).

Agora atenção: como algumas entregas deste PSet serão realizadas através de um repositório público no GitHub, você precisa criar um repositório padronizado para a disciplina. Faça o seguinte:

1. Crie um usuário no GitHub (ou use algum que você já possua);
2. Crie um repositório público com o nome `uvv_bd_1_TURMA`, por exemplo:
   - `uvv_bd_1_cc1m`
   - `uvv_bd_1_cc1mb`
   - `uvv_bd_1_cc2m`
   - `uvv_bd_1_si1n`
3. Documente o repositório utilizando arquivos Markdown (ver a seguir).
4. Crie, dentro do repositório, o subdiretório `pset1` e, dentro desse subdiretório, inclua um Markdown explicando e documentando para que serve o subdiretório.
5. Envie um e-mail para o professor, `abrantesasf@uvv.br`, com o endereço (URL) do seu repositório, especificando no e-mail: seu nome completo, matrícula, turma e URL do repositório GitHub.

2.3 GitHub Flavored Markdown

O **Markdown** é uma sintaxe para a formatação de arquivos escritos em texto puro que, posteriormente, podem ser convertidos para HTML.

A partir do Markdown puro (https://daringfireball.net/projects/markdown/) o GitHub adicionou algumas funcionalidades sintáticas adicionais e criou o **GitHub Flavored Markdown**, que é uma versão do Markdown específica para documentar páginas e repositórios no GitHub.

Sua próxima tarefa é aprender a utilizar arquivos Markdown para documentar tudo que você está fazendo durante a resolução deste PSet: você deve documentar o repositório, documentar os subdiretórios, documentar respostas, enfim, tudo que você escrever nos repositórios (exceto os códigos-fontes e demais arquivos de um projeto, como o leiame.txt, etc.) deverá estar no formato Markdown.

Aprenda sobre o GitHub Flavored Markdown em: https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/about-writing-and-formatting-on-github

Formate o arquivo "`README.MD`" de seu projeto e, usando o que você aprendeou sore o GitHub Markdown, ajuste o título, faça um cabeçalho com seu nome, o nome do professor e da monitora, e crie uma descrição geral.

2.4 Interfaces gráficas para Git e GitHub

O poder do Git está na linha de comando: se você dominar o uso do Git pela linha de comando, terá sempre à disposição tudo que você precisa para usar o Git e o GitHub com eficiência.

Entretanto, alguns usuários iniciantes podem se beneficiar de interfaces mais "amigáveis"[7], tais como:

- GitHub Desktop (https://desktop.github.com): é uma interface gráfica gratuita, para Windows e Mac, desenvolvida pelo próprio GitHub;
- TortoiseGit (https://tortoisegit.org): é uma "extensão" gratuita para o explorador de arquivos do Windows e cria uma integração muito interessante entre o Git, GitHub e Windows Explorer;
- GitKraken (https://www.gitkraken.com): é uma interface gráfica comercial (mas tem uma versão grauita), para Windows, Linux e Mac. É considerada por muitos como uma das melhores interfaces para o Git e GitHub;
- SmartGit (https://www.syntevo.com/smartgit): é uma interface gráfica comercial (não tem versão gratuita), para Windows, Linux e Mac. Também é considerada como uma das melhores interfaces para o Git e GitHub.

As interfaces citadas acima são apenas algumas das centenas de interfaces disponíveis. Lembre-se: você pode optar por uma interface gráfica mas o verdadeiro poder do Git está na linha de comando.

2.5 Questões discursivas sobre Git e GitHub

Responda às seguintes questões discursivas, nas folhas de papel almaço:

Questão 1: O que são sistemas de controles de versões? Por que são importantes?

Questão 2: Qual a diferença entre o Git e o GitHub? Como eles estão relacionados? É possível usar um sem o outro?

Questão 3: O Git é um sistema distribuído de controle de versões. O que significa isso?

[7] Há controvérsias se as interfaces gráficas são mais amigáveis mesmo… eu, por exemplo, acho que elas são confusas e mais atrapalham do que ajudam mas, confesso, que utilizo alguma interface gráfica de vez em quando.

3 Implementação de modelos lógicos nos SGBD

3.1 Tudo em scripts!

Nesta Seção você fará a implementação do projeto lógico que está nos Capítulos 5 e 6 do livro do Elmasri & Navathe, "Sistemas de Banco de Dados", 7ª edição (o livro de referência da disciplina) em, pelo menos dois SGBD: o PostgreSQL e o MariaDB/MySQL[8], mas atenção:

- Tudo que você fizer deve ser feito através de **scripts SQL** que você colocará em seu repositório Git/GitHub;
- Esses scripts devem estar prontos para serem rodados, **sem alteração nenhuma** por mim (para verificação e avaliação);
- Os scripts SQL devem ser **COMENTADOS ADEQUADAMENTE** para que eu saiba o que você está fazendo. Pesquise sobre a maneira correta de comentar um script SQL (há comentários de linha e comentários de blocos, você deverá usar os dois);
- Descubra como exeutar scripts SQL nos SGBD que você trabalhará, pois o modo de executar um script SQL no PostgreSQL é diferente do modo do Oracle, que é diferente do modo do MariaBD/MySQL e assim por diante. Cada SGBD executa scripts SQL de modo diferente;
- Você pode criar um único script SQL para cada SGBD ou pode dividir o trabalho em vários scripts. Neste último caso você deve nomear os scripts adequadamente e informar claramente a ordem na qual os scripts devem ser executados;
- Você pode utilizar softwares de projetos de bancos de dados (como o SQL Power Architect[9]) para gerar os scripts SQL, mas lembre-se: os scripts gerados por esses softwares geralmente precisam de ajustes finais para ficarem com tudo necessário; também precisam de receber comentários para indicar o que cada parte do script faz;
- Lembre-se: o seu repositório Git/GitHub é público, qualquer pessoa na internet terá acesso ao seu conteúdo. Não faça trabalhos desorganizados, bagunçados ou desleixados: o GitHub é uma excelente vitrine de projetos para recrutadores, e você não vai querer passar uma má impressão, correto?
- Você terá que pesquisar muita coisa específicas dos SGBD por conta própria, então não deixe para começar este PSet faltando poucos dias da data de entrega: não haverá tempo hábil para você terminar!

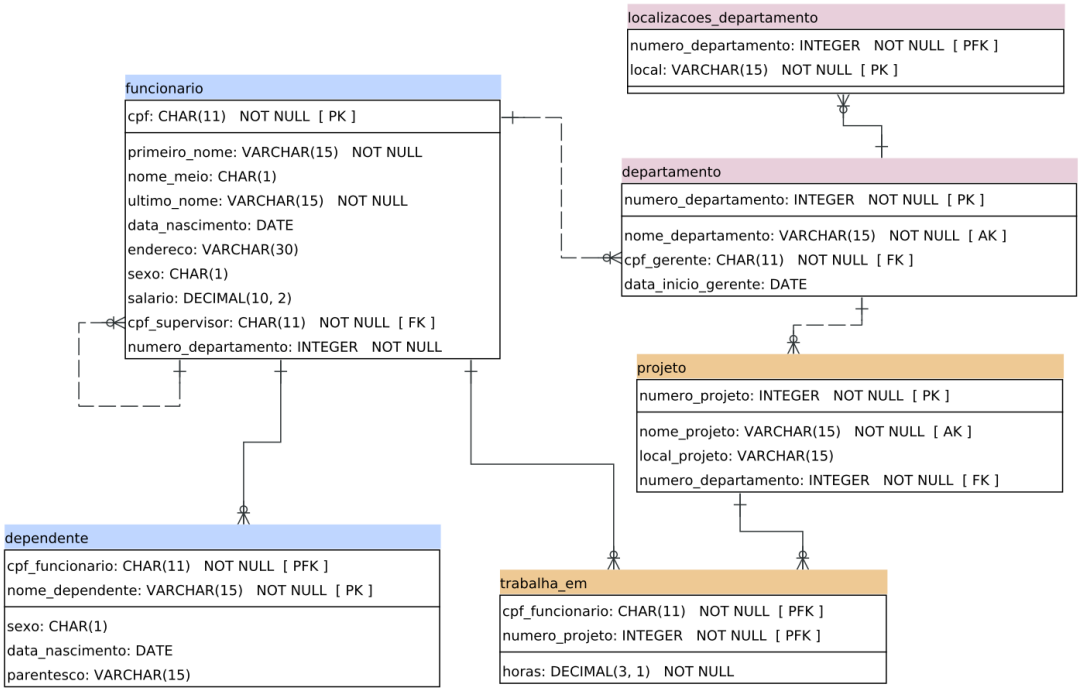
[8] Se você implementar nos quatro SGBD da máquina virtual, PostgreSQL, Oracle, MariaDB e MS SQL Server, aprenderá muito mais e poderá comparar as diferentes funcionalidades entre esses sistemas.

[9] http://www.bestofbi.com/page/architect_download_os

3.2 Modelo do Elmasri

O projeto lógico que você implementará no PostgreSQL e no MariaDB é o projeto que está descrito nos Capítulos 5 e 6 do Elmasri. Usando o SQL Power Architect eu criei o seguinte projeto para orientação:

Figura 1: Primeiro projeto



O diagrama acima está gravado em um arquivo de imagem que está em anexo a este PSet:

- `modelo-elmasri.png`; e

Além disso também está disponível junto com este PSet a documentação em HTML do projeto lógico. Este arquivo HTML foi gerado pelo SQL Power Architect e depois foi ajustado por mim (para acréscimo de informações e da figura do diagrama do projeto). O arquivo é:

- `elmasri.html`

Sua primeira tarefa é usar o SQL Power Architect para replicar exatamente o projeto lógico ilustrado acima. Atenção:

1. Use o diagrama do projeto (acima e nos arquivos em anexo) e também a documentação em HTML (em anexo) e recrie o projeto exatamente como está no SQL Power Architect;

2. Utilize todos os anexos fornecidos para reproduzir exatamente os comentários de cada tabela e de cada coluna. Quando você inclui no projeto os comentários adequados, esses comentários serão armazenados no catálogo junto aos metadados do banco de dados e, quando consultamos o catálogo, esses comentários serão exibidos, facilitando o entendimento da estrutura das tabelas e colunas;
3. Note que no projeto existem dois atributos com uma indicação que nós não vimos em aula (mas que vocês leram no Capítulo 6 do Elmasri), marcados como "[AK]". Esses atributos correspondem a chaves únicas, também conhecidas como chaves alternativas (por isso a sigla AK), e são implementados através de índices únicos nos atributos. Descubra como fazer isso no SQL Power Architect para incluir esses atributos corretamente no projeto;
4. Coloque o arquivo de seu projeto em SQL Power Architect (arquivo com a extensão final "architect") em seu repositório GitHub para avaliação.

3.3 Questões discursivas sobre o projeto lógico

Agora que você já conseguiu reproduzir o projeto lógico ilustrado na Figura 1, responda às seguintes questões discursivas nas folhas de papel almaço:

Questão 4: Existem pelo menos dois erros no projeto do Elmasri envolvendo as tabelas **funcionario** e **departamento**. Identifique quais são esses erros, explique o que está errado e conserte os erros se for necessário.

Questão 5: Alguma tabela do projeto representa um relacionamento do tipo N:N? Se sim, identifique a tabela e explique porque é um relacionamento N:N; Se não, explique porque não há relacionamentos N:N neste projeto.

Questão 6: Pelo projeto apresentado é possível que um determinado funcionário seja o gerente de um ou mais departamentos. Se eu quisesse impor uma regra que diz que um funcionário só pode ser gerente de, no máximo, um departamento, o que eu teria que fazer no projeto? O que eu teria que criar no banco de dados para impor essa restrição? Por quê?

Questão 7: Por que o relacionamento entre as tabelas **departamento** e **projeto** está representado como uma linha pontilhada? O que isso representa? Por que foi representado assim?

Questão 8: Qual é o único tipo de relacionamento que pode guardar dados? Por quê? Existe algum relacionamento assim no projeto do Elmasri?

Questão 9: Como explicar o relacionamento da tabela **funcionario** com ela mesma? É um erro? É correto? Por quê?

3.4 Implementação no PostgreSQL

O PostgreSQL (https://www.postgresql.org) é o SGBD relacional *open-source* mais avançado que existe hoje em dia. Apesar dele ser mais "difícil" que usar do que o MariaBD/MySQL, ele é excelente para estudar e aprender sobre bancos de dados pois ele têm funcionalidades avançadas que não são encontradas em muitos SGBD comerciais.

Sua tarefa é descobrir, por conta própria, como usar o PostgreSQL e como implementar o modelo lógico descrito na Seção 3.2 nesse SGBD. Leia a documentação no site do PostgreSQL, assista a vídeos no YouTube ou peça ajuda para os monitores. O importante é que você aprenda a utilizar o PostgreSQL!

Atenção: você pode utilizar algumas interfaces gráficas que já estão prontas na máquina virtual para trabalhar com o PostgreSQL e testar todos os comandos SQL que serão necessários, tais como o PgAdmin ou o DBeaver, mas o resultado final a ser publicado em seu repositório GitHub é um script SQL que eu devo executar em meu computador. Esse script SQL deve ter **todos os comandos**, na ordem correta e com comentários adequados, para que eu possa simplesmente rodar o script e avaliar o trabalho que você fez.

Estude a documentação do PostrgreSQL e faça o seguinte:

1. Vamos começar com uma tarefa de administração de sistemas Linux, a instalação do suporte para várias linguagens (pois assim o banco de dados a ser criado terá suporte para o Português). Abra um terminal Linux e torne-se superusuário (root) com o comando:

```
[computacao@dbserver ~]$ su -
Password:
```

   Depois de digitar a senha você verá o prompt de superusuário (root). Nesse prompt, digite o seguinte comando:

```
[>>> ROOT <<<@dbserver ~]# dnf install langpacks-en glibc-all-langpacks
```

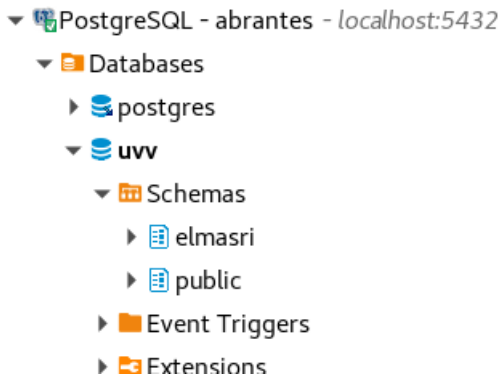
   Depois de algum tempo (e muitas mensagens esquisitas na tela) o suporte para várias linguagens será instalado no Linux. Nós temos que reiniciar o serviço do PostgreSQL e, para isso, ainda como root, execute o comando:

```
[>>> ROOT <<<@dbserver ~]# systemctl restart postgresql-14.service
```

   Agora você já pode fechar o terminal! Parabéns por ter concluído sua primeira tarefa de administração de sistemas Linux!

2. Crie um usuário específico para ser o "dono" do banco de dados que será criado (utilize o seu nome). Essa é uma boa prática de programação: nós nunca utilizamos os usuários administradores do SGBD para criar bancos de dados de produção. Consulte a documentação do comando `CREATE USER` e não se esqueça de que esse usuário deve ter a permissão de criar bancos de dados.
3. Depois de criar um usuários, faça login no PostgreSQL com esse usuário e crie o banco de dados "uvv" para a implementação do projeto do Elmasri. Consulte o comando `CREATE DATABASE` na documentação do PostgreSQL e utilize como parâmetros os seguintes valores:
   - nome do banco de dados: uvv
   - owner: o usuário que você criou
   - template: template0
   - encoding: UTF8
   - lc_collate: pt_BR.UTF-8
   - lc_ctype: pt_BR.UTF-8
   - allow_connections: true
4. Agora que você criou um banco de dados, faça uma conexão a esse banco de dados com o usuário que você criou (e que é o dono do banco de dados). Depois de se conectar, crie o esquema "elmasri" para armazenar todos os objetos que serão criados com a implementação do projeto lógico. Consulte a documentação do comando `CREATE SCHEMA` para saber como criar o esquema "elmasri" e autorizar o seu usuário (aquele que você criou no item 2) a utilizar esse esquema.

   Note que, no PostgreSQL, todo banco de dados recém-criado possui um esquema chamado "public". É nesse esquema público que os objetos do projeto lógico serão criados se não especificarmos algo diferente. Como queremos separar o projeto do Elmasri de outros projetos que faremos no futuro, dentro desse banco de dados chamado "uvv" criaremos o esquema "elmasri" para esse projeto. Se você fez tudo direito, deve ver alguma coisa assim (se estiver usando o DBeaver):



Por que essa confusão toda, com bancos de dados e esquemas? Para dar flexibilidade! Nós podemos nos referir a qualquer objeto de qualquer esquema dentro de um banco de dados através da notação de pontos da seguinte maneira: "banco.esquema.objeto". Por exemplo, se quisermos nos referir à tabela "funcionario" dentro do esquema "elmasri" no banco de dados "uvv", basta escrever: "uvv.elmasri.funcionario". Mas por que eu iria querer fazer isso? Porque utilizando um esqueme eu posso ter diferentes tabelas "funcionario" no mesmo banco de dados, cada uma em um esquema diferente!

5. Ajustando o esquema padrão: agora que você já criou o esquema "elmasri", perceba que o esquema que é utilizado por padrão no PostgreSQL é o esquema "public", justamente o que nós não queremos usar agora. Faça uma confirmação de que o esquema "public" realmente é o seu padrão com o comando "`SELECT CURRENT_SCHEMA();`". Você verá o esquema público como o padrão. Precisamos mudar isso para o esquema "elmasri", caso contrário todo objeto que for criado sem uma qualificação experssa do esquema será criado dentro do esquema público.

   No PostgreSQL a ordem dos esquemas é definida por um parâmetro chamado de `search_path`. Para você ver esse parâmetro, execute o comando "`SHOW SEARCH_PATH;`": você verá algo como: "$user", public. Temos que alterar o search_path para o usuário que você criou. Faça isso com o comando a seguir:

   ```
   SET SEARCH_PATH TO elmasri, "$user", public;
   ```

   Para confirmar que o search_path foi alterado, execute de novo o comando "`SHOW SEARCH_PATH;`": você verá que o esquema "elmasri" agora aparece na frente de todos e, por isso, ele será o esquema padrão.

   Mas o comando anterior tem uma desvantagem: ele é temporário, ou seja, se você sair do PostgreSQL e voltar depois, o esquema "elmasri" não continuará no search_path. Para isso temos que tornar essa mudança permanente para seu usuário, executando o comando abaixo:

   ```
   ALTER USER <nome do usuário>
   SET SEARCH_PATH TO elmasri, "$user", public;
   ```

   ATENÇÃO: algumas interfaces gráficas (DBeaver, RazorSQL, PgAdmin e outras) podem ter problemas para identificar o esquema padrão por causa de bugs (sim, isso já ocorreu antes). Então ao utilizar interfaces gráficas, tenha certeza de que você está conectado ao banco de dados correto e ao esquema correto! Utilize os comandos abaixo sempre que precisar:

   - `SHOW SEARCH_PATH;`
   - `SELECT CURRENT_SCHEMA();`
   - `SET SEARCH_PATH TO ...`
   - `ALTER USER x SET SEARCH_PATH TO ...`

6. Agora que você já está com o usuário, o banco de dados e o esquema prontos, é hora de implementar o projeto lógico do Elmasri! Volte a se conectar no banco de dados "uvv" e implemente todo o projeto da Figura 1 no PostgreSQL! Para isso:
    - Crie as tabelas com todos os campos, com os tipos de dados e restrições de NOT NULL corretas. ATENÇÃO: você deve incluir comentários nos metadados para descrever todas as tabelas e descrever todas as colunas em todas as tabelas. Consulte os comandos `COMMENT ON TABLE` e `COMMENT ON COLUMN` na documentação do PostgreSQL;
    - Crie as PK de cada tabela (cuidado com as PK compostas);
    - Crie os relacionamentos entre as tabelas (das FK para as PK);
    - Crie os índices únicos para os atributos que não podem receber valores repetidos (são os atributos com o qualificador [AK] no projeto lógico);
    - Crie quaisquer outras restrições que você achar importante na implementação do projeto, tais como: salário não pode ser negativo; sexo só pode ser F/M; horas trabalhadas nos projetos não podem ser negativas; etc.
7. Depois de implementar totalmente o projeto lógico, é hora de você inserir os dados que serão trabalhados futuramente neste PSet. Você terá que inserir todos os dados a seguir:

Figura 2: Tabela: funcionario

Primeiro_nome	Nome_meio	Ultimo_nome	Cpf	Data_nascimento	Endereco	Sexo	Salario	Cpf_supervisor	Numero_departamento
João	B	Silva	12345678966	09-01-1965	Rua das Flores, 751, São Paulo, SP	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	Rua Timbira, 35, São Paulo, SP	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	NULL	1

Figura 3: Tabela: departamento

Nome_ departamento	Numero_ departamento	Cpf_gerente	Data_inicio_ gerente
Pesquisa	5	33344555587	22-05-1988
Administração	4	98765432168	01-01-1995
Matriz	1	88866555576	19-06-1981

Figura 4: Tabela: localizacoes_departamento

Numero_de-partamento	Local
1	São Paulo
4	Mauá
5	Santo André
5	Itu
5	São Paulo

Figura 5: Tabela: projeto

Nome_projeto	Numero_ projeto	Local_projeto	Numero_ departamento
ProdutoX	1	Santo André	5
ProdutoY	2	Itu	5
ProdutoZ	3	São Paulo	5
Informatização	10	Mauá	4
Reorganização	20	São Paulo	1
Novosbenefícios	30	Mauá	4

Figura 6: Tabela: dependente

Cpf_funcionario	Nome_ dependente	Sexo	Data_nascimento	Parentesco
33344555587	Alicia	F	05-04-1986	Filha
33344555587	Tiago	M	25-10-1983	Filho
33344555587	Janaína	F	03-05-1958	Esposa
98765432168	Antonio	M	28-02-1942	Marido
12345678966	Michael	M	04-01-1988	Filho
12345678966	Alicia	F	30-12-1988	Filha
12345678966	Elizabeth	F	05-05-1967	Esposa

Figura 7: Tabela: trabalha_em

Cpf_funcionario	Numero_projeto	Horas
12345678966	1	32,5
12345678966	2	7,5
66688444476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98765432168	30	20,0
98765432168	20	15,0
88866555576	20	NULL

8. Depois de implementar totalmente o projeto lógico e de inserir os dados, prepare um script SQL com todos os comandos utilizados, na ordem correta, e envie esse script para seu repositório GitHub. Esse script deve incluir comandos para a criação de todas as tabelas, chaves e restrições, e também deve incluir todos os comandos de inserção de dados em todas as tabelas! Note que o script deve ser organizado, comentado e pronto para que eu execute em meu computador para verificar e avaliar seu trabalho!

3.5 Questões sobre a implementação no PostgreSQL

Agora que você já conseguiu implementar o projeto lógico no PostgreSQL, responda às seguintes questões discursivas nas folhas de papel almaço:

Questão 10: Qual a diferença entre **banco de dados**, **usuário** e **esquema** no PostgreSQL?

Questão 11: Por que um **esquema** é importante?

Questão 12: Se você não definir um esquema específico, onde os objetos do banco de dados (tabelas, relacionamentos, dados, etc.) serão gravados? Isso é bom ou ruim? Por que?

Questão 13: Agora que você já implementou o projeto no PostgreSQL, tem alguma sugestão de melhoria a fazer para o projeto? Como ele poderia ser melhorado?

3.6 Implementação no MariaBD/MySQL

Sua tarefa agora é fazer tudo o que você fez para o PostgreSQL novamente, só que usando o MariaBD/MySQL!

Descubra como criar o banco de dados "uvv", como criar seu usuário, como criar o esquema "elmasri" e como implementar o projeto lógico da Figura 1 no MariaBD (ou MySQL... a máquina virtual está com o MariaBD). É possível fazer no MariaBD/MySQL tudo o que o fizemos no PostgreSQL? É muito importante que você consulte a documentação do MariaBD (ou MySQL)!

Ao final de tudo, prepare um script SQL semelhante ao que você fez para o PostgreSQL, só que agora específico para o MariaBD/MySQL: o script deve contar todos os comandos, na ordem correta, para que eu crie as tabelas, restrições, chaves, relacionamentos e faça a inserção de dados de forma automática no meu computador para eu avaliar seu trabalho. Coloque esse script no seu repositório GitHub.

LEMBRE-SE: os scripts SQL devem ser organizados e bem comentados!

3.7 Implementação no Oracle

Que tal aproveitar o embalo e implementar o mesmo projeto no Oracle, o SGBD comercial mais avançado do mundo? Ele está prontinho na máquina virtual! Você só tem que descobrir como fazer as mesmas coisas no Oracle, mas atenção: o conceito de esquema do Oracle é totalmente diferente! Descubra na documentação e implemente. Depois prepare um script SQL e coloque no seu repositório GitHub também!

3.8 Questões finais

Depois de implementar o projeto lógico no PostgreSQL e no MariaBD (e, quem sabe, também no Oracle), responda às seguintes questões discursivas nas folhas de papel almaço:

Questão 14: Faça uma comparação dos SGBD que você utilizou. Quais as vantagnes e desvantagens de cada um? Quem tem a melhor documentação?

Questão 15: Como você acha que foi seu desempenho neste PSet? Como você foi buscando as informações necessárias e lendo as documentações dos SGBD? Você aproveitou a oportunidade e buscou ajuda dos monitores?

Até o próximo PSet pessoal!