MC536 – LAB 02

Usaremos o mesmo banco de dados do Lab 01. Caso tenha participado, pode pular para a seção de consultas. Caso não tenha participado, segue as instruções:

INTRUÇÕES

Baixar o arquivo `populate-database.sql` disponível em:
https://drive.google.com/file/d/1ueUp2tHfs9hGJqlfv5xQdQDmmgbTLQiS/view?usp=sharing

Ambiente para aula: sql.lab.ic.unicamp.br/adminer

sistema: PostgreSQL
servidor: localhost
usuário, *senha* e *database* são individuais e foram distribuídos em sala

Após efetuado login, clicar em *importar* → subir o arquivo `populate-database.sql` → executar script

O script criará as seguintes relações:

pessoa		
id	integer	Identificador da pessoa
nome	character varying	Primeiro nome
sobrenome	character varying	Sobrenome
ano_nascimento	integer	Ano de nascimento
nasceu	integer	Id da cidade em que nasceu
sexo	character(1)	Sexo da pessoa
ano_formacao_superior	integer	Ano em que a pessoa concluiu o ensino superior

cidade		
id	integer	Identificador da cidade
nome	character varying	Nome da cidade
pais	Character varying	Pais a qual a cidade pertence

viagem		
id_pessoa	integer	Identificador da pessoa
id_cidade	integer	Identificador da cidade destino
data	date	Data da viagem
custo	real	Custo total da viagem

As consultas a seguir estão acompanhadas de respostas. Tente resolve-las sem olhar as respostas. Estas consultas exploram vários tipos de junção, formuladas usando LEFT, RIGHT. INNER, NATURAL JOIN.

O mínimo básico a entender está em
https://www.w3schools.com/sql/sql_join.asp

Um exemplo de introdução mais didática sobre o assunto é
https://www.essentialsql.com/what-is-the-difference-between-an-inner-and-outer-join/

PARTE 1: CONSULTAS DE EXEMPLO

1. **Liste o nome das pessoas e o nome da cidade em que nasceram**

```
SELECT cidade.nome, pessoa.nome
FROM cidade, pessoa
WHERE nasceu = cidade.id
```

OU,

```
SELECT cidade.nome, pessoa.nome
FROM cidade INNER JOIN pessoa ON nasceu = cidade.id
```

OU

```
SELECT cidade.nome, pessoa.nome
FROM cidade JOIN pessoa ON nasceu = cidade.id
```

--- Quais as diferenças entre essas 3 formas de formular a consulta; você acha que o desempenho é diferente?

2. **Selecione o nome das pessoas que viajaram, a data e o custo de cada viagem**

```
SELECT nome, data, custo
FROM viagem JOIN pessoa ON id_pessoa = id
```

OU

```
SELECT pessoa.nome, data, custo
FROM pessoa NATURAL JOIN viagem AS v(id)
```

Porque foi necessário utilizar alias? É sempre necessário usar alias em casos de NATURAL JOIN? Qual a diferença entre INNER JOIN e NATURAL JOIN?

3. **Liste o id e nome de TODAS as cidades e o nome das pessoas que nasceram nelas.**

```
SELECT cidade.id, cidade.nome, pessoa.nome
FROM cidade LEFT OUTER JOIN pessoa ON nasceu = cidade.id
```

OU

```
SELECT cidade.id, cidade.nome, pessoa.nome
FROM cidade LEFT JOIN pessoa ON nasceu = cidade.id
```

Qual diferença entre as duas formas de consulta? Qual a diferença entre OUTER e INNER JOIN?

4. **Liste o nome de TODAS as pessoas e o nome da cidade em que nasceram**

```
SELECT pessoa.nome, cidade.nome
FROM cidade RIGHT OUTER JOIN pessoa ON nasceu = cidade.id
```

OU

```
SELECT pessoa.nome, cidade.nome
FROM cidade RIGHT JOIN pessoa ON
nasceu = cidade.id
```

O que há de diferente entre esta consulta e a que aparece no número 1?

5. **Liste o nome de TODAS as pessoas e o nome da cidade em que nasceram. As cidades em que ninguém nasceu também devem aparecer**

```
SELECT pessoa.nome, cidade.nome
FROM pessoa FULL OUTER JOIN cidade ON nasceu = cidade.id
```

6. **Para cada cidade, recupere seu nome e a quantidade de pessoas que nasceram nela**

```
SELECT cidade.nome, COUNT(nasceu)
FROM cidade LEFT JOIN pessoa ON nasceu = cidade.id
GROUP BY cidade.nome
```

7. **Para cada cidade em que nasceram mais de 2 pessoas, recupere o nome da cidade e a quantidade de pessoas que nasceram nela**

```
SELECT cidade.nome, COUNT(nasceu)
FROM pessoa JOIN cidade ON nasceu = cidade.id
GROUP BY cidade.nome
HAVING COUNT(*)>=2
```

PARTE 2: EXERCÍCIOS ADICIONAIS

1. **Para cada pessoa, recupere seu id, nome e a quantidade de cidades para onde já viajou. Ordene o resultado decrescentemente pela quantidade de viagens.**

Resultado esperado:

	id integer	nome character varying(80)	count bigint
1	1	Amanda	1
2	2	André	1
3	10	Diego	1
4	25	Simone	0
5	16	Mayara	0
6	11	Antônio	0
7	15	Franscisca	0
8	14	Mike	0
9	13	Clarisse	0
10	12	Josh	0
11	17	Mariane	0
12	5	Paula	0
13	51	Raul	0
14	3	Pedro	0
15	21	Ingrid	0
16	20	Manuela	0
17	50	Ana Paula	0
18	34	Julio	0
19	6	José	0
20	22	Emanuel	0
21	7	Paulo	0

2. **Obter nome e sobrenome das pessoas que já viajaram para a cidade natal de Manuela (Consulta proposta por Clara Pompeu)**

Resultado esperado: `Diego Oliveira`

3. **Liste o nome de TODAS as pessoas bem como custo e data de cada viagem que ela realizou.**

Resultado esperado:

	nome character varying(80)	custo real	data date
1	Diego	2000	2020-05-01
2	Amanda	30000	2015-04-10
3	André	5000	2018-12-01
4	Manuela		
5	Simone		
6	Antônio		
7	Mariane		
8	Julio		
9	Josh		
10	Franscisca		
11	Clarisse		
12	Ingrid		
13	Paula		
14	Ana Paula		
15	Raul		
16	José		
17	Mayara		
18	Emanuel		
19	Pedro		
20	Mike		
21	Paulo		

4. **Obter os nomes e sobrenomes de cada par de pessoas em que o ano de nascimento da primeira pessoa seja maior ou igual ao ano de formação superior da segunda pessoa (Consulta proposta por Guilherme Gama)**

Resultado esperado:

	nome character varying(80)	sobrenome character varying	ano_nascimento integer	nome character varying(80)	sobrenome character varying	ano_formacao_superior integer
1	Amanda	Silva	1987	Antônio	Silva	1975
2	Paula	Andrade	1990	Antônio	Silva	1975
3	Paulo	Batista	1987	Antônio	Silva	1975
4	Josh	Smith	1978	Antônio	Silva	1975
5	Franscisca	Sousa	1981	Antônio	Silva	1975
6	Mariane	Ramos	2000	Antônio	Silva	1975
7	Manuela	Andrade	2010	Josh	Smith	2005
8	Manuela	Andrade	2010	Antônio	Silva	1975
9	Manuela	Andrade	2010	José	Antunes	2009
10	Julio	Reis	1985	Antônio	Silva	1975
11	André	Sousa	1981	Antônio	Silva	1975
12	Diego	Oliveira	1993	Antônio	Silva	1975
13	Mayara	Santos	1990	Antônio	Silva	1975
14	José	Antunes	1985	Antônio	Silva	1975
15	Ana Paula	Batista	1989	Antônio	Silva	1975

5. **Obter o nome das cidades para as quais há viagens e o custo médio de se viajar para estas cidades, ordenado do mais caro ao mais barato. (Consulta proposta por Guilherme Gama (adaptada))**

Resultado esperado:

	nome character varying	custo_medio double precision
1	Nova York	30000
2	Berlim	5000
3	Rio Branco	2000

6. **Faça uma projeção dos gastos futuros com viagens: Obtenha a soma dos custos de viagens que ocorrerão a partir de hoje (utilize `current_date` para retornar o dia de hoje) (Consulta proposta por Guilherme Gama)**

7. **Crie um relacionamento chamado <mora> entre Pessoa e Cidade. Obs: uma pessoa mora numa cidade apenas, enquanto que numa cidade podem morar várias pessoas. Dica: Utilize o comando ALTER TABLE**

Como funciona ALTER TABLE? O que isto pode ocasionar nas estruturas de controle do SGBD?

8. **Insira os dados correspondentes ao relacionamento <mor>a com as seguintes informações (Dica: utilize o comando UPDATE):**
   Pessoa 1 mora na cidade 7
   Pessoa 2 mora na cidade 9
   Pessoa 3 mora na cidade 1
   Pessoa 5 mora na cidade 3
   Pessoa 7 mora na cidade 6
   Pessoa 12 mora na cidade 2
   Pessoa 13 mora na cidade 3
   Pessoa 15 mora na cidade 7
   Pessoa 17 mora na cidade 2
   Pessoa 20 mora na cidade 7
   Pessoa 34 mora na cidade 6

9. **Para cada cidade listar o nome daquelas que moram nela mas não nasceram lá**

10. **Crie um auto-relacionamento N x N em Pessoa, chamado `<conhece>`. Dica: Utilize o comando CREATE TABLE**

Analise este comando para entender qual a chave, e porque é N x N

11. **Insira dados no relacionamento `conhece` com as seguintes informações (Dica: utilize o comando `INSERT`):**

12. **Obter o id das pessoas que conhecem alguém que mora em Berlim**

Resultado esperado: `2 e 14`

13. **Selecione o custo das viagens feitas para uma cidade onde o viajante conhece alguém que mora lá**

Resultado esperado: `5000`

14. **Das pessoas que fizeram viagens internacionais, selecione nome, pais onde mora e pais para onde viajou.**

Resultado esperado:

	nome character varying(80)	pais_mora character varying	pais_destino character varying
1	Amanda	Alemanha	Estados Unidos da América
2	André	Espanha	Alemanha

PARTE 3: CRIACAO DE VISOES

Em aula, você ainda vai ver o que é uma visão, para que serve e como funciona.

Crie uma visão chamada <PEOPLE> da seguinte forma:

```
CREATE VIEW people AS
     SELECT * FROM pessoa
```

Qual o esquema de people?

Qual o esquema da visao <newpeople> abaixo?

```
CREATE VIEW newpeople AS
     SELECT *
     FROM pessoa
     WHERE id > 12
```

Qual o conteúdo de people e de newpeople?

Experimente atualizar people, da seguinte forma:

a) Consulte as relações pessoa, people e newpeople

```
(SELECT * FROM …)
```

b) Atualize pessoa, para id=13, indicando que o valor de mora passa a ser 999

```
UPDATE pessoa SET mora=999 WHERE id = 13
```

o que acontece? (consulte o conteudo de pessoa)

o que acontece com as visoes people e newpeople?

c) Tente agora atualizar valores via visoes, o que acontece com a tabela pessoa? E porque?