CONSULTAS EM BANCO DE DADOS

Alessandra Maciel Paz Milani

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Projeção em SQL

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir a operação de projeção em álgebra relacional.
- Exemplificar a operação de projeção em banco de dados relacional.
- Implementar a operação de projeção em SQL.

Introdução

Você já deve ter escutado que, além dos conceitos do modelo de dados para definir a estrutura do banco de dados, será necessário um conjunto básico de operações para manipulá-lo, conhecido, no modelo relacional, como álgebra relacional (ELMASRI; NAVATHE, 2011). Por conseguinte, há diferentes operações fundamentais na álgebra relacional, como seleção, projeção, produto cartesiano, união, etc.

Neste capítulo, você conhecerá a operação de projeção em álgebra relacional, identificará a operação de projeção em banco de dados relacionais por meio de diversos exemplos conceituais e compreenderá como implementar a operação de projeção em SQL no SGBD PostgreSQL.

1 Projeção em álgebra relacional

É importante lembrar que a álgebra relacional oferece um alicerce formal para as operações do modelo relacional, por meio dos quais o usuário conseguirá especificar as solicitações de recuperação para conjuntos ou relações. Note, ainda, que "[...] as entradas e as saídas de uma consulta são relações [...]" (RAMAKRISHNAN; GEHRKE, 2011, p. 84).

Elmasri e Navathe (2011) apontam que as operações da álgebra relacional podem ser divididas em dois grupos: um inclui o conjunto de operações da teoria de conjunto da matemática (união, intersecção, diferença de conjunto e produto cartesiano), e o outro consiste em operações desenvolvidas especificamente para banco de dados relacionais, como projeção e seleção. Além disso, uma vez que a projeção trabalha apenas com um conjunto de entrada, ela é classificada como uma operação relacional unária, ou seja, trata-se de uma função com somente uma variável de entrada.

Em seu dicionário de banco de dados relacionais, Date (2016) define simplificadamente projeção de como: deixe a relação **r** ter atributos chamados **A1**, **A2**, ..., **An**. Então, e somente então, a expressão **r**{**A1**, **A2**, ..., **An**} denota a projeção de **r** em {**A1**, **A2**, ..., **An**} e retorna a relação com o cabeçalho {**A1**, **A2**, ..., **An**} e o corpo constituído por todas as tuplas **t**, de modo que exista uma tupla em **r** que tenha o mesmo valor para os atributos **A1**, **A2**, ..., **An**, como **t** tem.

Outra maneira de formalizar a operação de projeção, representada pela letra grega *pi* (π), seria utilizarmos a seguinte expressão:

$$\pi_{\text{<Lista de atributos>}} \text{ (Relação)}$$

Então, por exemplo, considere a seguinte relação:

```
Funcionarios = {matricula, nome, endereco, data _ nascimento,
estado _ civil}
```

E esta instância:

```
Funcionarios = {1001, Ana, Rua Florida, 12/05/1970, Casado,
               1002, Carlos, Rua Tulipa, 22/06/1990, Solteiro,
                 1003, João, Avenida Flor, 01/02/1975, Casado,
                1004, Rubens, Rua Violeta, 02/10/1980, Casado}
```

Suponha que seja necessário buscar uma relação que apresente a `matricula`, o `nome` e o `estado_civil` dos funcionários — a formalização dessa operação poderá ser representada como:

$$\pi_{\text{matrícula, nome, estado civil}} (\text{Funcionários})$$

Por fim, o retorno dessa operação de projeção será a relação `Funcionarios2={matricula, nome, estado_civil}` contendo as seguintes tuplas:

```
Funcionarios2 = {1001, Ana, Casado,
                 1002, Carlos, Solteiro,
                 1003, João, Casado,
                 1004, Rubens, Casado}
```

Outra analogia para entender esse conceito reside no fato de que a projeção produzirá como saída um subconjunto vertical do conjunto sob análise. Além disso, intuitivamente, você pode visualizar essa estrutura de dados como uma tabela; portanto, podemos dizer que a operação de projeção (π) selecionará colunas de interesse dessa tabela, como exemplificado na Figura 1.

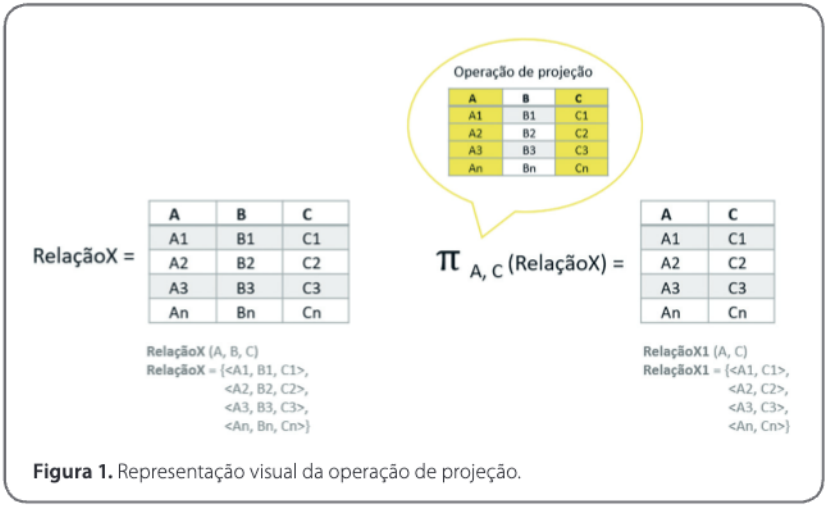


Figura 1. Representação visual da operação de projeção.

A seguir, observaremos como se comportam as buscas com projeção de dados por exemplos em tabelas.

2 Exemplos de projeção em banco de dados relacional

Vamos conferir mais alguns exemplos para a operação de projeção em banco de dados, iniciando pelo caso de uma relação ou um conjunto, que aqui chamaremos de tabela `aluno`, que contém dados de alunos. Nessa tabela, há seis atributos, ou colunas, listados a seguir:

1. `id_aluno` = número identificador do aluno;
2. `nome` = nome de registro do aluno;
3. `nome_pref` = nome preferencial do aluno;
4. `data_nasc` = data de nascimento;
5. `sit_matr` = situação de matrícula, onde `0` = Não Ativo e `1` = Ativo;
6. `email` = endereço eletrônico do aluno.

Considere um caso no qual você precisa enviar uma mensagem de alerta por *e-mail* para todos os alunos. Com base no que já vimos neste capítulo, como poderíamos escrever uma expressão para essa consulta? E qual seria o resultado dessa busca considerando o exemplo da instância apresentada no Quadro 1?

Quadro 1. Tabela para representação do conjunto de aluno

id_ aluno	nome	nome_ pref	data_ nasc	sit_ matr	email
27001	João Gustavo Rico	João Gustavo	12/04/1990	0	Jg_rico@ mail.com
27043	Luís Tadeu Marciano	Marciano	26/08/1999	1	tadeum@ mail.com
27023	Otávio Nunes Santos	Nunes	05/05/1998	1	o_nunes@ mail.com

(Continua)

(Continuação)

Quadro 1. Tabela para representação do conjunto de aluno

id_aluno	nome	nome_pref	data_nasc	sit_matr	email
28056	Marta Azevedo	Marta	10/06/1992	1	m.azevedo@mail.com
28943	Michele Lima Toledo	Mi	22/12/1995	0	mi_lt@mail.com

A resposta para essa questão está apresentada na Figura 2, na qual há uma representação visual para ilustrar o problema discutido, relacionado à tabela `aluno`. Na sequência, são apresentados a formulação da operação de projeção em álgebra relacional e o resultado esperado para essa operação de projeção, indicado por meio de uma nova tabela, de coluna única (`email`), na qual aparece o endereço eletrônico para os alunos que constavam na tabela original.

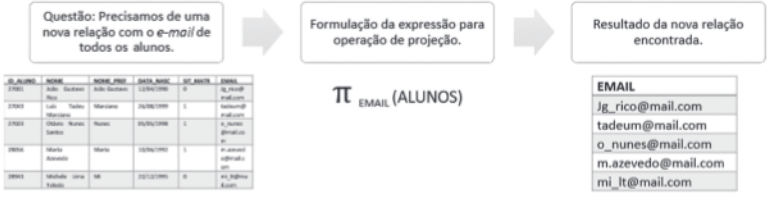


Figura 2. Representação visual para a operação de projeção da coluna email para a tabela aluno.

Agora, suponha que você tem interesse em saber também o nome preferencial pelo qual o aluno gostaria de ser chamado. O que seria preciso alterar nessa operação de projeção para considerar esse atributo adicional? Simples! Basta adicionarmos o nome do atributo correspondente, quando a nova expressão fica da seguinte maneira:

$$\pi_{\text{nome\_pref, email}} (\texttt{aluno})$$

A nova relação resultante para essa operação de projeção está apresentada no Quadro 2.

Quadro 2. Resultado da operação de Projeção na tabela aluno, considerando o novo atributo a ser apresentado

NOME_PREF	EMAIL
João Gustavo	jg_rico@mail.com
Marciano	tadeum@mail.com
Nunes	o_nunes@mail.com
Marta	m.azevedo@mail.com
Mi	mi_lt@mail.com

Outro exemplo seria considerar um conjunto de dados para clientes de uma loja de vendas pela internet. Em uma tabela chamada `cliente_compra`, 10 atributos foram criados para armazenar os dados dos clientes e de suas compras. Considerando os elementos da relação apresentada no Quadro 3, você consegue identificar qual seria a operação de projeção aplicada para gerar esse retorno?

Quadro 3. Tabela para representação do conjunto cliente_compra resultante de uma operação de projeção

cp	nome_cli	dt_compra	qtde_itens	valor_total
12345678987	Gustavo Rico	10/03/2020	3	50
34125678943	Tadeu Marciano	10/03/2020	6	180
34561278956	Jardel Nunes	11/03/2020	2	600
34567891278	Joaquim Azevedo	11/03/2020	4	150
12563478990	Paulo Lima	12/03/2020	10	200

Para obtermos o retorno apresentado no Quadro 3, precisaremos formalizar uma operação de projeção que indique nosso interesse em apresentar as mesmas colunas que aparecem na tabela resultante; portanto, deveremos escrever o nome dessas cinco colunas: `cpf_cli`, `nome_cli`, `dt_compra`, `qtde_itens`, `valor_total`. Ainda, precisaremos informar qual é a tabela de origem para recuperar esses dados, ou seja, incluir o nome da `cliente_compra`. Dessa maneira, podemos formular a seguinte expressão em álgebra relacional, lembrando-se de iniciar com a letra pi, já que ela representa a operação de projeção.

$$\pi_{\text{CPF\_CLI, NOME\_CLI, DT\_COMPRA, QTDE\_ITENS, VALOR\_TOTAL}}(\text{CLIENTE\_COMPRA})$$

3 Implementação de projeção em SQL

Agora que você já está familiarizado com a operação de projeção, veremos como ela pode ser implementada na linguagem prática para o modelo relacional. Vale ressaltar que a linguagem SQL (Structured Query Language) foi desenvolvida com base em linguagens de consultas formais, entre elas a álgebra relacional. Assim, você consegue identificar qual seria a parte correspondente

em uma instrução SQL para a operação de projeção? Na Figura 3, observamos como se daria a implementação SQL correspondente ao primeiro exemplo deste capítulo, da relação de funcionários.

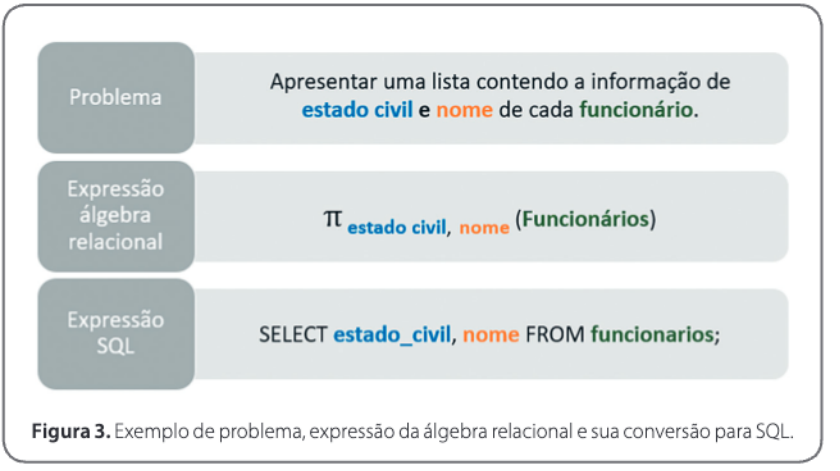


Figura 3. Exemplo de problema, expressão da álgebra relacional e sua conversão para SQL.

A partir desse momento, usaremos em nossos exemplos o Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL (POSTGRE, 2020). Para apresentarmos as execuções dos comandos de consulta e dos respectivos resultados, usaremos o *software* gráfico pgAdmin, disponibilizado gratuitamente junto ao pacote de instalação do PostgreSQL.

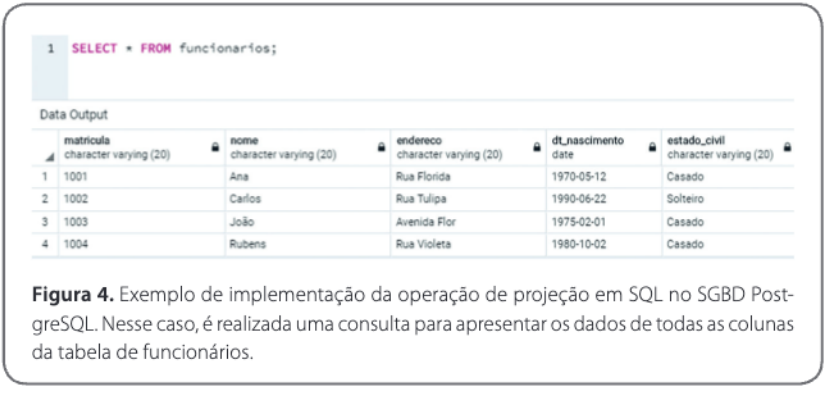
Note ainda que, para esse exemplo funcionar, precisamos ter criado antes uma tabela `funcionarios` e, então, inserir alguns elementos de exemplo. Nesse processo, ajustaremos os nomes dos atributos para remover acentos ortográficos e espaços em branco entre as palavras. Assim, você pode considerar o seguinte comando para a criação da tabela de funcionários:

```
CREATE TABLE public.funcionarios (matricula varchar(20), nome
varchar(20), endereco varchar(20), dt_nascimento date, es-
tado_civil varchar(20));
```

Depois, alguns dados de funcionários podem ser adicionados à tabela criada. No nosso exemplo, adicionamos informações para quatro funcionários, conforme os comandos indicados a seguir:

```
INSERT INTO public.funcionarios(matricula, nome, endereco,
dt_nascimento, estado_civil)VALUES ('1001', 'Ana', 'Rua Flo-
rida', to_date('12-05-1970','DD-MM-YYYY'), 'Casado');
INSERT INTO public.funcionarios(matricula, nome, endereco,
dt_nascimento, estado_civil)VALUES ('1002', Carlos, 'Rua Tu-
lipa, to_date('22-06-1990','DD-MM-YYYY'), 'Solteiro');
INSERT INTO public.funcionarios(matricula, nome, endereco,
dt_nascimento, estado_civil)VALUES ('1003', 'João', 'Avenida
Flor', to_date('01-02-1975','DD-MM-YYYY'), 'Casado');
INSERT INTO public.funcionarios(matricula, nome, endereco,
dt_nascimento, estado_civil)VALUES ('1004', 'Rubens', 'Rua
Violeta', to_date('02-10-1970','DD-MM-YYYY'), 'Casado');
```

Ao final desse processo, temos disponível uma tabela de funcionários com colunas e linhas conforme indicado na Figura 4: no topo, aparece o comando SQL (iniciado com o comando `SELECT`), utilizado para a operação de projetar todas as colunas e todas as possíveis linhas dessa tabela de funcionários; e, abaixo (com o título Data Output), o resultado da execução desse comando, em uma visão tabular, com todas as informações contidas na tabela `funcionarios`.



```
1  SELECT * FROM funcionarios;
```

Data Output

	matricula character varying (20)	nome character varying (20)	endereco character varying (20)	dt_nascimento date	estado_civil character varying (20)
1	1001	Ana	Rua Florida	1970-05-12	Casado
2	1002	Carlos	Rua Tulipa	1990-06-22	Solteiro
3	1003	João	Avenida Flor	1975-02-01	Casado
4	1004	Rubens	Rua Violeta	1980-10-02	Casado

Figura 4. Exemplo de implementação da operação de projeção em SQL no SGBD PostgreSQL. Nesse caso, é realizada uma consulta para apresentar os dados de todas as colunas da tabela de funcionários.

Agora, confira na Figura 5, o detalhe para a implementação da operação de projeção em SQL do nosso problema de exemplo, o mesmo apresentado na Figura 3. No topo da imagem, aparece a linha correspondente ao comando SQL, para que sejam projetadas apenas duas colunas específicas, e, logo abaixo, consta a informação resultante dessa operação de projeção.

```
1  SELECT estado_civil, nome FROM funcionarios;
```

Data Output

	estado_civil character varying (20)	nome character varying (20)
1	Casado	Ana
2	Solteiro	Carlos
3	Casado	João
4	Casado	Rubens

Figura 5. Implementação da operação de projeção em SQL no SGBD PostgreSQL. Nesse caso, é realizada uma consulta para apresentar as informações contidas em duas colunas específicas: `estado_civil` e `nome` da tabela `funcionarios`.

Em SQL, a projeção corresponderá à cláusula `SELECT`: você poderá escolher listar apenas alguns atributos da sua tabela, informando, assim, o nome de cada um deles na cláusula `SELECT`, como no exemplo da Figura 5, ou, então, indicar que tem a intenção de projetar todos os atributos por meio do asterisco, como no exemplo da Figura 4.

Fique atento

Apesar de a operação de projeção dispor de uma estrutura simples, na prática a maneira como você implementa o seu comando `SELECT` poderá impactar o desempenho da sua consulta.

Por exemplo, o fato de você indicar que sua consulta deve projetar todas as colunas de uma tabela usando o asterisco "`*`" no lugar de apenas indicar aquelas colunas que sejam realmente o seu objeto de interesse pode resultar em um maior tempo de processamento para o SGBD retornar a consulta. Entre os motivos para isso, está a premissa de que, ao termos mais colunas projetadas, potencialmente teremos mais dados em escopo e, logo, mais "custo" para o SGBD.

Evidentemente, essa questão dependerá muito de uma série de configurações do SGBD em uso, das características da sua tabela (número de colunas, tipos de dados, número de linhas, se índices de busca foram criados, etc.), da combinação da operação de projeção com outras operações relacionais e do problema em questão a que você precisa atender.

Então, para alguns casos nos quais a consulta envolve uma tabela com poucas colunas e poucos dados, dificilmente o fato de solicitar a busca por todas as colunas causará um impacto perceptível aos usuários. Contudo, essa questão poderá ficar bastante evidente em cenários nos quais grandes volumes de dados estão sendo manipulados.

Assim, como uma boa prática de desenvolvimento de consultas, principalmente quando da incorporação de uma solução a um sistema comercial, você deve considerar a indicação das colunas de interesse (somente). E, caso necessário, retornar todas colunas; então, ainda seria preferível que você indicasse explicitamente o nome de cada uma delas na sua cláusula `SELECT` (em vez de usar o "`*`").

A otimização de consulta é um tópico de estudo mais avançado, pois requerer o estudo prévio de outros temas, em especial as demais operações da álgebra relacional. Mas, se estiver curioso, pode consultar o capítulo 19 do livro de Elmasri e Navathe (2011), que está indicado na lista de referências deste capítulo.

Além disso, considerando que você faça uma consulta para retornar somente o atributo `estado_civil`, você verá como resultado 4 linhas, uma para cada tupla; assim, aparecerão informações repetidas, como mostra a Figura 6. Esse não seria o resultado esperado para álgebra relacional. Então, por que isso aconteceu? Por questões de desempenho, o SGBD está configurado para realizar a consulta com o plano mais simples. Dessa maneira, ele não avalia duplicidades sem que seja requerido.

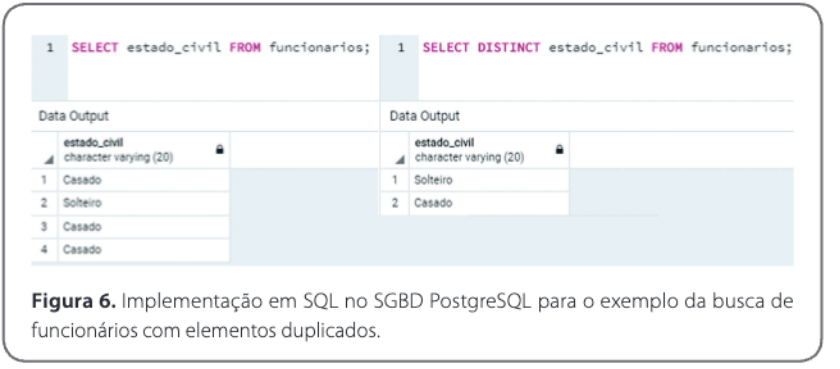


Figura 6. Implementação em SQL no SGBD PostgreSQL para o exemplo da busca de funcionários com elementos duplicados.

Para que as duplicidades não ocorram, você poderá utilizar o comando `DISTINT` logo após o `SELECT` e antes da lista de atributos. Por meio dessa especificação, o SGBD entenderá que precisa descartar duas ou mais linhas com os mesmos valores para as colunas em projeção, como exemplificado na Figura 7.

Figura 7. Implementação em SQL no SGBD PostgreSQL para o exemplo da busca de funcionários com o uso de `DISTINCT` para que não retornem elementos duplicados.

Observe que a ordem com que a colunas são informadas na cláusula `SELECT` será a mesma em que as informações serão apresentadas no conjunto resultante (veja o exemplo da Figura 5). No caso da escolha por todas as colunas, a ordem apresentada será a mesma na qual a tabela foi criada (veja o exemplo da Figura 4).

Por fim, diferentes operações podem ser combinadas com a projeção, desde uma operação básica, como a seleção, que adicionará a cláusula `WHERE` ao comando SQL para que a consulta retorne apenas elementos que atendam à determinada condição, até operadores aritméticos, como um contador de ocorrências de elementos para uma coluna em especial. Contudo, por aqui encerramos a introdução à operação de projeção da álgebra relacional, com seus respectivos exemplos aplicados a tabelas e implementados em SQL.

Referências

DATE, C. J. *The new relational database dictionary*: terms, concepts, and examples. Sebastopol: O'Reilly, 2016.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

POSTGRE. [*Site*]. 2020. Disponível em: https://www.Postgre.org/. Acesso em: 25 maio 2020.

RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011. *E-book*.

Leituras recomendadas

MACHADO, F. N. R. *Projeto e implementação de banco de dados*. 3. ed. São Paulo: Érica, 2014.

MANNINO, M. V. *Projeto, desenvolvimento de aplicações e administração de banco de dados*. 3. ed. Porto Alegre: AMGH, 2008.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS