BANCO DE DADOS

Roni Francisco Pichetti

SOLUÇÕES EDUCACIONAIS INTEGRADAS

Álgebra relacional

Objetivos de aprendizagem

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

- Identificar as operações de álgebra relacional associadas à linguagem SQL.
- Praticar a utilização de operadores de álgebra relacional e consultas em SQL.
- Reconhecer as diferentes implementações dos operadores de álgebra relacional em bancos de dados comerciais.

Introdução

Os bancos de dados foram criados para garantir a integridade das informações utilizadas em diferentes sistemas de informações, seja a nível operacional ou de gestão das informações. Nesse sentido, os sistemas de gerenciamento de bancos de dados relacionais (SGBDRs) são baseados em conceitos da álgebra relacional e tratam da manipulação e recuperação de dados em expressões algébricas e matemáticas. Isso possibilita a construção de consultas organizadas e eficientes.

Neste capítulo, você vai estudar a álgebra relacional e a sua relação com a linguagem de consulta estruturada (SQL, do inglês *structured query language*). Para isso, você vai aprender sobre os operadores de álgebra relacional e vai verificar exemplos da criação de consultas em SQL utilizando esses operadores em bancos de dados para sistemas comerciais.

1 Operadores de álgebra relacional em SQL

A **álgebra relacional** é um conjunto básico de operações para o modelo relacional de banco de dados. Essas operações possibilitam que o usuário especifique as recuperações básicas como expressões da álgebra relacional. Portanto, o resultado de uma expressão de álgebra relacional será uma relação que representa o resultado de uma consulta em um banco de dados. A álgebra relacional é importante, pois oferece uma base formal para as operações do modelo relacional. Assim, ela é utilizada para o aumento da eficiência de consultas nos módulos de otimização e processamento de consultas, que fazem parte dos SGBDRs, conforme lecionam Elmasri e Navathe (2011).

As operações de álgebra relacional podem ser divididas em dois grupos: um grupo que inclui um conjunto de operações da teoria de conjuntos da matemática (união, intersecção, diferença de conjunto e produto cartesiano), e outro que trata de operações desenvolvidas para bancos de dados relacionais (seleção, projeção, junção, divisão, renomeação e alteração). Algumas solicitações em bancos de dados comuns não podem ser realizadas com as operações originais da álgebra relacional; por esse motivo, algumas operações foram criadas especificamente para expressá-las. Entre esses casos estão as funções de agregação, que podem resumir os dados das tabelas, e os tipos adicionais de junção e união, que são a junção externa e a união externa (ELMASRI; NAVATHE, 2011). Para Machado (2014), existem outros tipos de classificação desses mesmos operadores, como quanto à sua origem (fundamentais, derivados e especiais) e quanto ao número de relações (unitários ou binários).

As consultas em álgebra relacional utilizam os operadores para descrever um procedimento passo a passo para obter um resultado. Nesse caso, tanto as entradas quanto as saídas se tornam relações. Isso quer dizer que uma consulta é avaliada utilizando as instâncias de cada relação de entrada e a instância da relação de saída. Ao definir a álgebra relacional, é possível referenciar os campos pela sua posição, em vez de pelo seu nome, para que não seja necessário referenciar todos os resultados intermediários. Porém, ao utilizar os nomes dos campos, as consultas se tornam mais legíveis (RAMARKRISHNAN; GEHRKE, 2011).

Diferentes operações da teoria de conjuntos são utilizadas para mesclar os elementos de dois conjuntos, como união, intersecção e diferença de conjunto (ou subtração). Essas são operações em que cada uma é aplicada a dois conjuntos de linhas. Quando essas operações são utilizadas nos bancos de dados relacionais, as duas relações que serão analisadas ou comparadas precisam possuir o mesmo tipo de tuplas — o que é chamado de **compatibilidade de união ou de tipo**. Essa compatibilidade é verificada pela coincidência de quantidade de atributos e de domínio das tuplas (ELMASRI; NAVATHE, 2011).

Nesse sentido, para compreender melhor o conceito das operações união, intersecção e subtração, vejamos a seguir a comparação entre duas relações, *Q* e *R*, com cada um dos operadores.

- A **união** (∪) de *Q* e *R*, portanto, $Q \cup R$, é uma relação que inclui todas as tuplas que estão em *Q* ou em *R* ou em *R* e em *Q*, em que as duplicadas são eliminadas.
- A **intersecção** (∩) de *Q* e *R*, portanto, $Q \cap R$, é uma relação somente com as tuplas que estão tanto em *Q* quanto em *R*, as tuplas em comum.
- A **subtração** (−), representada por $Q - R$, trata basicamente das tuplas que estão em *Q*, mas não estão em *R*.

Veja, a seguir, um exemplo que ilustra essas relações.

Exemplo

(Produtos *Q*) ∪ (produtos *R*):

Nome produto
Água sem gás
Refrigerante
Suco

U

Nome produto
Gelo
Água com gás

=

Nome produto
Água sem gás
Refrigerante
Suco
Gelo
Água com gás

(Produtos *Q*) ∩ (produtos *R*):

Nome produto
Água sem gás
Refrigerante
Suco

∩

Nome produto
Gelo
Água sem gás
Suco

=

Nome produto
Água sem gás
Suco

(Produtos *Q*) – (Produtos *R*):

Nome produto
Água sem gás
Refrigerante
Suco

–

Nome produto
Gelo
Água sem gás
Suco

=

Nome produto
Refrigerante

Pode-se perceber que as operações de união e intersecção são **comutativas**; isso quer dizer que, independentemente da ordem da relação, o resultado será o mesmo. Então $Q \cup R = R \cup Q$ e $Q \cap R = R \cap Q$. Já a operação de subtração é **não comutativa**, pois a diferença é verificada da primeira para a segunda tabela. Portanto, $Q - R \neq$ de $R - Q$. Em SQL, existem três operações que realizam as mesmas funções das operações de conjunto união, intersecção e subtração, que são, respectivamente, `UNION`, `INTERSECT` e `EXCEPT` (ELMASRI; NAVATHE, 2011).

A operação de conjunto **produto cartesiano**, ou produto cruzado, é representada por ×. Quando utilizada, suas relações não precisam ser compatíveis na união. Por ser binária, produz um novo elemento, combinando cada tupla de uma relação com cada tupla da outra relação. Dessa forma, resulta em uma nova tabela, com todas as combinações possíveis entre os elementos das tabelas originais. Ela possui maior utilidade quando é seguida por uma seleção que combine valores de atributos das relações que a compõem. Em SQL, a operação produto cartesiano pode ser realizada empregando-se a opção `CROSS JOIN` em tabelas juntadas (ELMASRI; NAVATHE, 2011). Veja, a seguir, um exemplo da aplicação dessa operação.

Exemplo

(Aluno *Q*) × (Matéria *R*):

Aluno	Matrícula
Danilo	1234
José	7654
Ana	4321

×

Matéria	Código
Física	9999
Inglês	8888

=

Aluno	Matrícula	Matéria	Código
Danilo	1234	Física	9999
Danilo	1234	Inglês	8888
José	7654	Física	9999
José	7654	Inglês	8888
Ana	4321	Física	9999
Ana	4321	Inglês	8888

Entre os operadores da álgebra relacional está o de **seleção**, representado por σ (letra grega sigma), que é utilizado para selecionar as linhas de uma relação. Há também o operador de **projeção**, representado por π (letra grega Pi), que é empregado para projetar colunas. Assim, o operador seleção especifica as linhas a serem mantidas utilizando uma condição de seleção, que geralmente é uma expressão booleana (comparação entre dois valores) de termos (RAMARKRISHNAN; GEHRKE, 2011). Segundo Elmasri e Navathe (2011), a operação de seleção pode ser considerada um filtro, que mantém apenas as linhas que satisfazem uma determinada condição, restringindo as demais. Entre os operadores de seleção utilizados, estão: {=, <, >, ≠, ≤, ≥}.

Enquanto a operação de seleção escolhe algumas linhas e descarta outras, a operação de projeção seleciona colunas e descarta outras. Por esse motivo, o operador de projeção é utilizado quando são necessários apenas determinados atributos de uma relação. Assim, o resultado de uma projeção pode ser visualizado como uma parte vertical da relação. A operação de projeção remove linhas que estiverem duplicadas; portanto, o resultado dessa operação é um conjunto de linhas distintas. Essa eliminação das duplicatas necessita de uma classificação ou de outra técnica, o que aumenta o processamento (ELMASRI; NAVATHE, 2011).

A operação de **junção** é indicada por ⋈ e é utilizada para combinar tuplas relacionadas de duas relações em uma única tupla. Essa operação é importante, pois possibilita processar mais de um relacionamento entre as relações. A operação de junção pode ser definida como uma operação de produto cartesiano aliada a uma operação de seleção. Em uma junção, apenas as combinações de tuplas que contemplarem a condição junção previamente

definida aparecem no resultado. Assim, cada combinação de tupla na qual a condição de junção é considerada verdadeira é incluída no resultado como uma única tupla combinada. Portanto, nem todas as informações das relações participantes da operação de junção são preservadas no resultado, visto que as tuplas que não correspondem à condição não são combinadas (ELMASRI; NAVATHE, 2011).

A operação de **divisão** (÷) é utilizada em um tipo específico de consulta que pode ocorrer em banco de dados. Por exemplo, pesquisar os nomes dos funcionários que trabalham em todos os projetos em que "André Silveira" trabalha. Para isso, primeiro, deve ser recuperada a lista dos números de projeto em que "André Silveira" trabalha. Depois, deve-se criar uma relação que ligue a matrícula do funcionário a um projeto. E, por fim, deve-se aplicar a divisão nas duas relações, o que resulta na matrícula dos funcionários desejados (ELMASRI; NAVATHE, 2011).

Para facilitar a conexão com operações complexas, pode ser útil renomear os atributos nas relações intermediárias e de resultado, ou o nome da relação. Para isso, é utilizada a operação **renomear** (ρ). Por exemplo, pode-se renomear a relação comprador para cliente, com a expressão: $\rho_{comprador}^{(cliente)}$, em que "comprador" é o nome novo e "cliente" é o nome antigo da relação (ELMASRI; NAVATHE, 2011).

2 Operações de álgebra relacional e consultas SQL

A álgebra relacional é muito importante para compreender os tipos de solicitações que podem ser definidas em um banco de dados relacional. Ela também é útil para o processamento e a otimização das consultas em um SGBDR. Porém, as operações da álgebra relacional são consideradas muito técnicas pela maioria dos usuários de sistemas de gerenciamento de banco de dados (SGBDs) comerciais, visto que uma consulta em álgebra relacional, para ser escrita, necessita de uma sequência de operações, que, quando forem executadas, produzem o resultado esperado. Dessa forma, o usuário necessita especificar como e em que ordem executar as operações de consulta. Já a SQL possibilita uma interface de linguagem declarativa de nível mais alto, para que o usuário especifique apenas qual deve ser o resultado, enquanto o SGBD gerencia a otimização e as decisões de como executar a consulta (ELMASRI; NAVATHE, 2011).

A SQL possui muitos recursos baseados em álgebra relacional; porém, sua sintaxe é mais fácil de ser utilizada do que a linguagem formal da álgebra relacional (ELMASRI; NAVATHE, 2011). Por esse motivo, nesta parte do capítulo, serão apresentados alguns exemplos dos operadores de álgebra relacional e a sua **correspondência em SQL**, para uso em bancos de dados. Para iniciar, veja o Quadro 1, que traz operadores de álgebra relacional relacionados às cláusulas e operadores utilizados em SQL.

Quadro 1. Operadores de álgebra relacional e respectivas cláusulas e operadores em SQL

Álgebra relacional	SQL
Projeção	`SELECT`
Produto cartesiano	`FROM`
Seleção	`WHERE`
União	`UNION`
Renomeação	`AS`
Intersecção	`INTERSECT`
Subtração	`MINUS ou EXCEPT`
Junção	`JOIN`

Fonte: Adaptado de Elmasri e Navathe (2011).

Veja os exemplos do *box* a seguir.

Exemplo

O exemplo a seguir trata da operação de seleção em uma tabela de colaboradores em que o código do departamento deve ser 4, e o salário, maior do que R$ 30.000,00 (ELMASRI; NAVATHE, 2011).

Operador de seleção (σ):

$$\sigma_{\text{CodigoDepto=4 AND Salario>30.000}}(\text{COLABORADOR})$$

corresponde à consulta SQL:

```
SELECT *
FROM COLABORADOR
WHERE CodigoDepto=4 AND Salario>30.000;
```

O próximo exemplo relaciona o operador de projeção, também na tabela de colaboradores. Nesse caso, são especificadas duas colunas, `Sexo` e `Salario`, sendo que o resultado não pode possuir duplicatas. Para isso, utiliza-se a palavra-chave `DISTINCT`. Caso não fosse empregada, as duplicatas poderiam não ser eliminadas, o que não é permitido nesse caso da álgebra relacional formal (ELMASRI; NAVATHE, 2011).

Operador de projeção (π):

$$\pi_{\text{Sexo, Salario}}(\text{COLABORADOR})$$

corresponde à consulta SQL:

```
SELECT DISTINCT Sexo, Salario
FROM COLABORADOR;
```

Por fim, temos um exemplo da operação **renomear** em uma consulta na tabela `COLABORADOR`. Nessa consulta, está sendo requisitado o nome e o salário dos colaborares que trabalham no departamento 4. Com a operação de renomeação, serão alterados os nomes da relação criada e dos atributos.

Operador de renomeação (ρ):

$$\text{CodDepto4\_Colab} \leftarrow \sigma_{\text{CodigoDepto=4}}(\text{Colaborador})$$

$$\rho\ \text{Resultado}_{\text{(Pnome, Unome, Sal)}} \leftarrow \pi_{\text{Primeiro\_nome, Ultimo\_nome, Salario}}(\text{CodDepto4\_Colab})$$

corresponde à consulta SQL:

```
SELECT C.Pnome AS Primeiro _ nome, C.Unome AS Ultimo _ nome,
C.Sal AS Salario
FROM COLABORADOR AS C
WHERE CodigoDepto=4;
```

Percebe-se que, em SQL, a renomeação é possibilitada pelo emprego dos *alias* ou apelidos, utilizando `AS`, o que é útil na realização de consultas sobre muitos atributos, por exemplo.

3 Uso de operadores de álgebra relacional em bancos de dados comerciais

O **modelo relacional** como um todo se baseia diretamente na álgebra relacional, o que permite que os dados sejam armazenados de forma consistente, com redundância reduzida e com a integridade das informações garantida. O modelo relacional adquiriu uma importância cada vez maior desde sua criação, em 1970, por Edgar Frank Codd, sendo implementado em milhões de bancos de dados comerciais dos mais diversos tipos de negócios. Pode-se dizer que os pontos fortes do modelo relacional são a integridade e a baixa redundância das informações e, por esses motivos, ainda será a melhor solução para bancos de dados comerciais por muito tempo (AMARAL, 2016).

Ainda sobre a aplicação de operadores de álgebra relacional em SQL, a seguir, estão relacionados exemplos de operadores de união, intersecção e subtração com seus respectivos correspondentes em SQL: `UNION`, `INTERSECT` e `EXCEPT`.

Aplicação de operadores matemáticos de conjuntos

Veja, a seguir, exemplos de operadores matemáticos de conjuntos (união, intersecção e subtração).

União

Listar colaboradores que trabalham no departamento 4 ou que supervisionam um colaborador que trabalhe nesse mesmo departamento:

$$\text{Depto4} \leftarrow \sigma_{\text{CodigoDepto=4}}^{\text{(Colaborador)}}$$

$$\text{Result1} \leftarrow \pi_{\text{Matricula\_Colaborador}}^{\text{(Depto4)}}$$

$$\text{Result2} \leftarrow \pi_{\text{Matricula\_Supervisor}}^{\text{(Depto4)}}$$

$$\text{Resultado} \leftarrow \text{Result1} \cup \text{Result2}$$

corresponde à consulta SQL:

```
SELECT COLABORADOR.Matricula _ Colaborador
FROM COLABORADOR
WHERE CodigoDepto=4;
UNION
SELECT COLABORADOR.Matricula _ Supervisor
FROM COLABORADOR
WHERE CodigoDepto=4;
```

Intersecção

Listar colaboradores que trabalham no departamento 4 e que supervisionam um colaborador que trabalhe nesse mesmo departamento:

$$\text{Depto4} \leftarrow \sigma_{\text{CodigoDepto=4}}^{\text{(Colaborador)}}$$

$$\text{Result1} \leftarrow \pi_{\text{Matricula\_Colaborador}}^{\text{(Depto4)}}$$

$$\text{Result2} \leftarrow \pi_{\text{Matricula\_Supervisor}}^{\text{(Depto4)}}$$

$$\text{Resultado} \leftarrow \text{Result1} \cap \text{Result2}$$

corresponde à consulta SQL:

```
SELECT COLABORADOR.Matricula _ Colaborador
FROM COLABORADOR
WHERE CodigoDepto=4;
INTERSECT
SELECT COLABORADOR.Matricula _ Supervisor
FROM COLABORADOR
WHERE CodigoDepto=4;
```

Subtração

Listar colaboradores que trabalham no departamento 4, mas que não supervisionam um colaborador que trabalhe nesse mesmo departamento:

$$\text{Depto4} \leftarrow \sigma_{\text{CodigoDepto=4}}^{\text{(Colaborador)}}$$

$$\text{Result1} \leftarrow \pi_{\text{Matricula\_Colaborador}}^{\text{(Depto4)}}$$

$$\text{Result2} \leftarrow \pi_{\text{Matricula\_Supervisor}}^{\text{(Depto4)}}$$

$$\text{Resultado} \leftarrow \text{Result1} - \text{Result2}$$

corresponde à consulta SQL:

```
SELECT COLABORADOR.Matricula _ Colaborador
FROM COLABORADOR
WHERE CodigoDepto=4;
EXCEPT
SELECT COLABORADOR.Matricula _ Supervisor
FROM COLABORADOR
WHERE CodigoDepto=4;
```

Link

Você viu ao longo deste capítulo diferentes conceitos e exemplos de expressões em álgebra linear utilizando intersecção e subtração. Acessando o *link* a seguir, você pode conferir conceitos adicionais sobre os operadores de conjunto `INTERSECT` e `EXCEPT`, que podem ser empregados em diferentes versões de bancos de dados em SQL.

http://bit.ly/2TEjBt6

Os próximos exemplos tratam da comparação dos operadores de junção e divisão, com suas correspondentes consultas em SQL, tendo em vista que as operações de junção e divisão são consideradas binárias, pois dependem da relação entre duas tabelas (RAMARKRISHNAN; GERHRKE, 2011).

Junção

Listar o nome do supervisor e de cada departamento:

$$\text{Sup\_Depto} \leftarrow \text{Departamento} \bowtie_{\text{NumSupervisor} = \text{NumColaborador}}^{(\text{Colaborador})}$$

$$\text{Resultado} \leftarrow \pi_{\text{NomeDepto, PrimeiroNome, UltimoNome}}^{(\text{Sup\_Depto})}$$

corresponde à consulta SQL:

```
SELECT NomeDepto, PrimeiroNome, UltimoNome
FROM (COLABORADOR JOIN DEPARTAMENTO ON
NumSupervisor=NumColaborador);
```

Divisão

Listar todos os projetos nos quais o colaborador com o código 10 trabalha:

$$\text{Colab10} \leftarrow \sigma_{\text{NumColaborador=10'}}^{(\text{Colaborador})}$$

$$\text{Projs\_C10} \leftarrow \pi_{\text{Num\_Proj}}^{(\text{Trabalha\_em} * \text{Andre})}$$

$$\text{NumColabs\_NumProjs} \leftarrow \pi_{\text{NumColaborador, Num\_Projeto}}^{(\text{Trabalha\_em})}$$

$$\text{NumColabs} \leftarrow \pi\ \text{NumColabs\_NumProjs} \div \text{Projs\_C10}$$

$$\text{Resultado} \leftarrow \pi_{\text{PrimeiroNome, UltimoNome}}^{(\text{NumColabs} * \text{Colaborador})}$$

corresponde à consulta SQL:

```
SELECT COLABORADOR.PrimeiroNome, COLABORADOR.UltimoNome
FROM COLABORADOR
WHERE NumColaborador = 10
MINUS
SELECT NumColaborador, NumProjeto
FROM TRABALHA _ EM
WHERE NumColaborador = 10;
```

Referências

AMARAL, F. *Introdução a ciência de dados*: mineração de dados e big data. Rio de Janeiro: Alta Books, 2016.

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

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

RAMARKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011.

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*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS