VOLTAR

Álgebra Relacional - Seleção e Projeção

Apresentar os conceitos de álgebra relacional envolvendo as operações de seleção e projeção.

NESTE TÓPICO

Marcar tópico

SQL – Structured Query Language

SQL é atualmente a principal linguagem utilizada para realizar consultas e manipulações de dados em Sistemas de Gerenciamento de Bancos de Dados Relacionais.

A primeira versão da linguagem foi apresentada pela IBM em 1974 com o nome Structured English Query Language (Sequel) e disponibilizada para um protótipo de banco de dados relacional denominado SEQUEL-XRM.

Em 1977, a IBM lançou um novo protótipo denominado SYSTEM/R e, ao final deste, havia desenvolvido uma linguagem que permitia acesso fácil a múltiplas tabelas, uma linguagem de quarta geração (4GL), que passou a ser denominada de SQL (Structured Query Language).

Alguns anos depois, em 1979, um grupo de desenvolvedores que havia participado do projeto SYSTEM/R fundou uma empresa – a Relational Software Inc. – responsável pelo lançamento do primeiro Sistema de Gerenciamento de Banco de Dados Relacional comercialmente viável. Esta empresa mais tarde teve o seu nome alterado para Oracle.

A linguagem SQL é composta basicamente por quatro subconjuntos:

- DDL – Data Definition Language

- DML – Data Manipulation Language
- DQL – Data Query Language
- DCL – Data Control Language

Durante o seu curso você terá outras disciplinas que abordarão com detalhes cada um dos comandos utilizados pelos quatro subconjuntos da SQL.
No entanto, você terá agora a oportunidade de conhecer alguns comandos relacionados ao que foi apresentado nas aulas anteriores quando foram listadas as operações da álgebra relacional.

Seleção

Conforme observado na aula 15, esta operação, indicada pela letra grega Σ (sigma), produz uma nova relação ou tabela apenas com as tuplas (linhas) da primeira relação que satisfazem a uma determinada condição (também chamada de predicado).

Tomando-se como exemplo a seguinte tabela, para efetuar a **seleção** do funcionário cujo ID_FUNC = 102, devemos utilizar a expressão:

σ ID_FUNC=102 (FUNCIONARIO)

ID_FUNC	NOME_FUNC	CARGO
101	Antonio Alves	Analista Pleno
102	Beatriz Bernardes	Analista Pleno
103	Claudio Cardoso	Analista Senior
104	Daniela Dantas	Analista Senior

A expressão acima pode ser escrita na linguagem SQL da seguinte forma:
`SELECT * FROM FUNCIONARIO WHERE ID_FUNC = 102;`

Observa-se, portanto, que nesta linguagem utilizamos a palavra reservada **SELECT** no lugar da letra grega Σ (sigma). O símbolo ***** é utilizado quando queremos que sejam apresentados os dados de todas as colunas da tabela.

A preposição **FROM** é utilizada antes de informarmos o nome da tabela consultada: **FUNCIONARIO**.

Na sequência, temos a condição, isto é, queremos que sejam apresentados os dados do funcionário cujo ID_FUNC é igual a 102. Expressamos isso com o seguinte predicado: **WHERE ID_FUC = 102**. A execução do comando acima produzirá o seguinte resultado:

ID_FUNC	NOME_FUNC	CARGO
102	Beatriz Bernardes	Analista Pleno

Projeção

Muitas vezes não desejamos que sejam apresentados todos os dados que satisfazem a determinada condição. No exemplo considerado, uma hipótese seria apresentar apenas o NOME do funcionário cujo ID_FUNC é igual a 102. Neste caso, devemos **projetar** a coluna. Na SQL informamos os nomes das colunas que queremos projetar logo após o comando SELECT. Observe: **SELECT NOME_FUNC FROM FUNCIONARIO WHERE ID_FUNC = 102;**

A execução do comando acima produzirá o seguinte resultado:

NOME_FUNC
Beatriz Bernardes

Para projetar mais de uma coluna, separamos seus nomes utilizando vírgulas: **SELECT NOME_FUNC, CARGO FROM FUNCIONARIO WHERE ID_FUNC = 102;**

A execução do comando acima produzirá o seguinte resultado:

NOME_FUNC	CARGO
Beatriz Bernardes	Analista Pleno

Mas qual comando deve ser utilizado para projetar TODOS os nomes dos funcionários? Neste caso, basta omitir a condição expressa no final. Observe: **SELECT NOME_FUNC FROM FUNCIONARIO;**

A execução do comando acima produzirá o seguinte resultado:

NOME_FUNC
Antonio Alves
Beatriz Bernardes
Claudio Cardoso
Daniela Dantas

Produto cartesiano

O produto cartesiano de duas tabelas ou relações é uma terceira tabela contendo todas as combinações possíveis entre as tuplas ou linhas da primeira e as tuplas da segunda tabela.

Observe novamente o exemplo apresentado na aula 16, o produto cartesiano das tabelas GRUPO_SP e GRUPO_RJ:

GRUPO_SP
Corinthians
Palmeiras
Santos
São Paulo

GRUPO_RJ
Botafogo
Flamengo
Fluminense
Vasco

GRUPO_SP X GRUPO_RJ	
Corinthians	Botafogo
Corinthians	Flamengo
Corinthians	Fluminense
Corinthians	Vasco
Palmeiras	Botafogo
Palmeiras	Flamengo
Palmeiras	Fluminense
Palmeiras	Vasco
Santos	Botafogo
Santos	Flamengo
Santos	Fluminense
Santos	Vasco
São Paulo	Botafogo
São Paulo	Flamengo
São Paulo	Fluminense
São Paulo	Vasco

A representação, conforme a álgebra relacional, do produto cartesiano acima é a seguinte:

(GRUPO_SP) X (GRUPO_RJ)

Quando trabalhamos com a SQL, o produto cartesiano de duas tabelas é obtido por meio da operação denominada **CROSS JOIN**. Observe:

```
SELECT TIME_SP, TIME_RJ FROM GRUPO_SP CROSS JOIN GRUPO_RJ;
```

A operação de junção, conforme veremos a seguir, é uma derivação do produto cartesiano.

Junção (normal)

A operação de junção utiliza as operações de seleção e produto cartesiano para produzir uma combinação entre as tuplas (linhas) de uma tabela com as tuplas correspondentes de outra tabela que obedecem a uma condição.

Observe novamente o exemplo apresentado na aula 18:

DEPARTAMENTO	
CODDEPT	NOMEDEPT
D1	Engenharia
D2	Comercial

FUNCIONARIO		
DFUNC	NOMEFUNC	CODDEPT
101	Antonio Alves	D2
102	Beatriz Bernardes	D1
103	Claudio Cardoso	D2
104	Daniela Dantas	D1

A **junção** entre as tabelas DEPARTAMENTO e FUNCIONARIO deve ser representada, conforme notação da álgebra relacional, da seguinte forma:

σ **DEPARTAMENTO.CODDEPT = FUNCIONARIO.CODDEPT (DEPARTAMENTO X FUNCIONARIO)**

O resultado produzido será o seguinte:

CODDEPT	NOMEDEPT	IDFUNC	NOMEFUNC	CODDEPT
D1	Engenharia	102	Beatriz Bernardes	D1
D1	Engenharia	104	Daniela Dantas	D1
D2	Comercial	101	Antonio Alves	D2
D2	Comercial	103	Claudio Cardoso	D2

A expressão acima pode ser escrita na linguagem SQL da seguinte forma: **SELECT * FROM DEPARTAMENTO, FUNCIONARIO WHERE DEPARTAMENTO.CODDEPT = FUNCIONARIO.CODDEPT;**

Junção (natural)

A **junção natural**fornece o mesmo resultado da junção normal, mas sem a repetição de valores das colunas comuns às duas tabelas.

A junção natural entre as tabelas DEPARTAMENTO e FUNCIONARIO é representada, conforme notação da álgebra relacional, da seguinte forma: (DEPARTAMENTO |x| FUNCIONARIO)

Quando utilizamos a SQL, a junção natural de duas tabelas é obtida por meio da operação denominada **NATURAL INNER JOIN**. Observe: **SELECT * FROM DEPARTAMENTO NATURAL INNER JOIN FUNCIONARIO;**

A figura a seguir demonstra o resultado da operação de **junção natural** entre as duas tabelas anteriores (DEPARTAMENTO e FUNCIONARIO):

CODDEPT	NOMEDEPT	IDFUNC	NOMEFUNC
D1	Engenharia	102	Beatriz Bernardes
D1	Engenharia	104	Daniela Dantas
D2	Comercial	101	Antonio Alves
D2	Comercial	103	Claudio Cardoso

Observamos nesta aula como utilizar a SQL (Structured Query Language) para realizar as seguintes operações da álgebra relacional: seleção, projeção, produto cartesiano e junção (normal e natural). Na última aula desta disciplina abordaremos as outras operações da álgebra relacional: união, diferença e intersecção.

Referências

CHEN, Peter. *Modelagem de dados*: a abordagem entidade-relacionamento para projeto lógico. São Paulo: Makron Books, 1990.

DATE, C. J. *Introdução a sistemas de banco de dados*. Rio de Janeiro: Campus, 1991.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. 4. ed. São Paulo: Pearson Addison Wesley, 2005.

HEUSER, Carlos Alberto. *Projeto de banco de dados*. Porto Alegre: Sagra Luzzatto, 2004.

MULLER, Robert J. *Projeto de Banco de Dados*: usando UML para modelagem de dados. São Paulo: Berkeley Brasil, 2002.

SETZER, Valdemar W.; SILVA, Flávio Soares Corrêa da. *Banco de dados*: aprenda o que são, melhore seu conhecimento, construa os seus. São Paulo: Edgard Blücher, 2005.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Sistema de banco de dados*. 3. ed. São Paulo: Makron Books, 1999.

Avalie este tópico

ANTERIOR
Álgebra Relacional - Junção e Divisão

Índice

PRÓXIMO
União, Diferença e Intersação