MODELAGEM E DESENVOLVIMENTO DE BANCO DE DADOS

Pedro Henrique Chagas Freitas

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linguagem de Consultas de Dados: DQL (*Data Query Language*)

Objetivos de aprendizagem

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

- Identificar a linguagem DQL.
- Exemplificar a linguagem DQL.
- Implementar a linguagem DQL.

Introdução

Neste capítulo, você vai estudar a modelagem e o desenvolvimento de banco de dados a partir da perspectiva da DQL (*Data Query Language*), observando as estruturas que a compõem, bem como exemplos de sua utilização e a sua implementação na consulta a bases de dados.

A DQL (*Data Query Language*) é uma linguagem de consulta utilizada na composição da SQL (*Structured Query Language*), que é uma linguagem estruturada e difundida para bancos de dados relacionais inspirada em álgebra relacional. A DQL será apresentada ao longo deste capítulo de forma holística, com uma abordagem de todas as características que vinculam essa linguagem, bem como sua implementação na modelagem e no desenvolvimento de bancos de dados.

Conceituando a linguagem DQL

A linguagem DQL (*Data Query Language*) é a única composição da linguagem SQL que possui um único comando, o SELECT. O SELECT é uma instrução SQL pertencente à DQL segundo a visão de alguns autores; para outros autores, o SELECT faz parte da linguagem DML (*Data Manipulation Language*), que também compõe o SQL. De qualquer forma, o comando SELECT é o mais

utilizado pelos usuários de sistemas gerenciadores de bancos de dados, tendo em vista que, a partir do SELECT, podemos realizar consultas aos bancos de dados, retornando dados, parâmetros, composições, dentre diversos tipos de consultas.

Por sua vez, a DQL, devido à característica de ter o comando SELECT em seu núcleo, torna-se essencial na compreensão da modelagem e no desenvolvimento de bancos de dados, já que os bancos de dados são projetados para receber consultas com diversos tipos de complexidades.

Fique atento

Apesar de alguns autores alocarem o SELECT como uma instrução DQL (Figura 1) e outros como uma instrução DML, na prática, isso é um mero formalismo, tendo em vista que a utilização do SELECT nas consultas (que é um dos fundamentos principais de acesso à base de dados) é o que realmente importa. Logo, não se prenda, neste caso, em definir se, ao realizar uma consulta, estamos manipulando os dados ou somente selecionando-os.

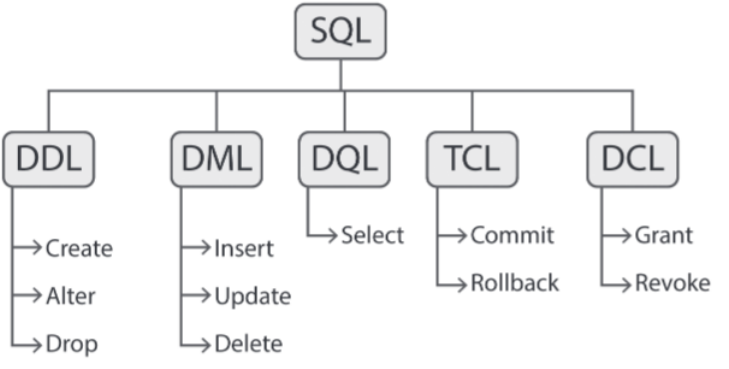

Figura 1. SQL: composições e comandos.

Dessa forma, assim como a qualidade do dado é essencial para o bom funcionamento dos bancos de dados, as consultas realizadas pelo comando SELECT do DQL são fundamentais na utilização dos bancos de dados pelos Sistemas Gerenciadores de Bancos de Dados (SGBDs).

O comando DQL SELECT permite ao usuário de um banco de dados realizar uma consulta a partir de uma especificação de parâmetro, também conhecido como *query* — uma vez que a linguagem se chama *Data Query Language*. A especificação da *query* apresenta a descrição do resultado de retorno desejado após a consulta.

É importante observar que a derivação dessa consulta é um retorno esperado, isto é, quando realizamos uma consulta em uma base de dados, estamos realizando uma consulta de algo em algum lugar, de modo que os parâmetros que envolvem a localização dos dados precisam apresentar exatidão conforme a complexidade da consulta. Veja o exemplo a seguir:

```
SELECT nome_campo FROM nome_tabela WHERE condição_esperada
```

O comando SELECT é realizado a partir de um parâmetro de localização inicial, ou seja, o nome do campo, aqui representado por: nome_campo. A localização desse campo existe para referenciar um dado, que, neste caso, está em uma tabela, por isso usamos FROM nome_tabela para designar a tabela onde está localizado o campo que contém o dado que desejamos consultar. Por fim, temos uma condição de retorno ou condição_esperada, que é uma restrição quanto ao retorno da consulta.

Na *Data Query Language,* é necessário compreender o dado na sua real localização e nas suas condições de retorno, ou seja, se você precisa consultar algo em algum lugar, deve dizer para o seu SGBD, a partir da linguagem DQL, o nome desse algo, onde ele está e qual é a restrição do retorno que deseja receber.

Saiba mais

É importante destacar que, no princípio da criação da DQL, era fundamental que a instrução SELECT não realizasse mudança nos dados do banco de dados após a consulta ser realizada, caso contrário, teríamos inconsistências nos dados que seriam manipulados. É por essa razão que alguns autores costumam definir o SELECT como uma instrução SQL pertencente ao DML, porque entendem que, ao consultar o dado, já se está manipulando esse dado. No entanto, a criação da DQL tem como princípio a não manipulação do dado a partir da consulta, isto é, o dado permanece puro, somente é consultado, sem receber alterações.

Todavia, não podemos deixar de dizer que isso é plenamente conceitual; logo, quando uma consulta é realizada na base de dados, pode ocorrer, ou não, a manipulação dos dados e um UPDATE. Entretanto, no contexto em que estamos abordando a DQL, esta não é a intenção inicial da consulta à base dados.

Exemplificando a linguagem DQL

A cláusula WHERE tem uma finalidade essencial na linguagem DQL no que se refere às consultas realizadas. É por meio dessa cláusula que se estabelecem as condições que a consulta deve obedecer para realizar o retorno dos valores especificados.

Caso não utilizássemos a cláusula WHERE, teríamos retorno de todos os registros de uma determinada tabela. Por exemplo:

```
SELECT * FROM Times;
```

Nesse caso, são retornados todos os registros da tabela "Times" sem que haja uma especificação sobre o que se deseja retornar. Obviamente, isso é possível, mas utilizamos parâmetros para retorno — no caso, as cláusulas exatamente, porque consultas extensas podem onerar os SGBDs, tendo em vista que a maioria desses sistemas não se dedica a uma única instrução de consulta. Assim, consultas sem definição de parâmetros podem criar cenários de grande dispêndio de tempo, de modo que o normal é que as consultas utilizem parâmetros de retorno. Por exemplo:

```
SELECT Nome_Time, Região_Time FROM Times;
```

Neste caso, é possível demonstrar que os únicos dados que deverão ser retornados após consulta da tabela "Times" são o "Nome do time" (Nome_Time) e a "Região do time" (Região_Time). Dentro desse contexto, quando inserimos a cláusula WHERE, podemos aplicar mais um filtro no retorno dos registros. Por exemplo, podemos trazer só o registro em que o nome do time é Corinthians.

```
SELECT Nome_Time, Região_Time FROM Times WHERE Nome_Time =
'Corinthians';
```

No caso, então, podemos usar vários tipos sintaxe para apoiar nossa consulta, dando a ela cada vez mais efetividade e complexidade. Por exemplo, podemos fazê-lo utilizando funções embarcadas na consulta, como o Distinct, que visa não permitir que valores semelhantes de uma mesma coluna sejam retornados. Observe o exemplo:

```
SELECT DISTINCT (Nome_Campo) FROM Nome_Tabela
```

Fique atento

Hoje existem diversos recursos na linguagem SQL para respaldar diferentes cenários de consulta por meio da linguagem DQL utilizando a instrução SELECT. Não cabe decorar esses recursos, como funções de ordenamento, distinção, cruzamento, etc., tendo em vista que cada tipo de consulta é único e vai pedir uma determinada regra para a realização da consulta.

Assim sendo, é completamente normal que os DBAs (*DataBase Administrator*), ou Administradores de Banco de Dados, realizem verificações na literatura SQL para validar o tipo de consulta que será realizado dependendo dos dados que se deseja consultar. Desse modo, não é preciso decorar tudo, o que é necessário saber é a lógica por trás das instruções de consulta e o funcionamento e a utilização do comando SELECT, que é o fundamento da DQL.

Implementando a DQL

Nesta implementação da linguagem DQL (Figura 2), vamos utilizar a instrução SELECT para realizar consultas na tabela CLIENTES já previamente populada com dados. Faremos as consultas inicialmente para retornar dados contidos em dois campos (CODIGO e NOME) utilizando uma cláusula de restrição de retorno nas nossas consultas, isto é, utilizaremos o WHERE para delimitar o parâmetro que deverá ser retornado da nossa tabela CLIENTE.

Implementação:

```
SELECT CODIGO, NOME FROM CLIENTES WHERE CODIGO = 10;
```

Neste caso, estamos fazendo uma consulta na tabela CLIENTES pedindo o retorno dos dados do campo CODIGO e NOME com a restrição de retorno, onde o CODIGO = 10.

CODIGO	NOME
10	JOAO

Figura 2. Implementação da DQL — Tabela CLIENTES.

Implementação:

```
SELECT CODIGO, NOME FROM CLIENTES WHERE UF = 'RJ'
```

Neste caso, estamos referenciando nossa tabela CLIENTE (Figura 2) para retornar, a partir de consulta com SELECT, dados pertencentes a dois campos — CODIGO e NOME —, com a restrição de que os campos devem ser retornados, onde a UF = RJ.

UF RJ

CODIGO	NOME
3	CARLOS

Figura 3. Implementação da DQL — Tabela CLIENTES com restrição.

Implementação:

```
SELECT CODIGO, NOME FROM CLIENTES WHERE CODIGO >= 100 AND
CODIGO <=500;
```

Neste caso, estamos consultando a tabela CLIENTES nos respectivos campos CODIGO e NOME com a restrição de retorno no CODIGO, sendo este > ou = 100 e <= 500. Logo, aqui, temos duas restrições de retorno que vão limitar a consulta. Temos, então, CODIGOS sendo retornados após e igual a 100 e antes e igual a 500 conforme pode-se ver na Figura 4.

CODIGO	NOME
100	PEDRO
101	RICARDO
102	LUIS
(...)	(...)
499	SOCRATES
500	CAIO

Figura 4. Implementação da DQL — Tabela CLIENTES com duas restrições.

Implementação:

```
SELECT CODIGO, NOME FROM CLIENTES WHERE UF = 'MG' OR UF = 'SP'
```

Neste caso, utilizamos uma restrição de escolha (OR) que representaria um "OU", isto é, ou um (MG) ou outro (SP). Logo, consultamos a tabela CLIENTES pedindo retorno nos campos CODIGO e NOME, onde UF = MG ou UF = SP (Figuras 5 e 6). Caso UF = MG, teremos CODIGO e NOME vinculados a UF MG; caso UF = SP, teremos CODIGO e NOME vinculados a SP.

UF MG

CODIGO	NOME
685	TIAGO
686	FILIPE

Figura 5. Implementação da DQL — Tabela CLIENTES e retornos.

UF SP

CODIGO	NOME
458	HIURY
459	CALEBE

Figura 6. Implementação da DQL — Tabela CLIENTES e retornos.

Assim, encerramos por aqui, lembrando que o DQL possui um único comando para consulta à base de dados, isto é, o SELECT, que é bastante utilizado na modelagem e no desenvolvimento de bancos dados. Não esqueça que cada consulta trará, em si mesma, suas necessidades e seus parâmetros.

Saiba mais

A normalização é um conceito muito pertinente e de essencial atenção nas consultas a bases de dados, tendo em vista que, ao utilizar o comando DQL SELECT em bases de dados desnormalizadas, podemos receber retornos desnecessários — além disso, pode existir um grande risco na perda de recurso de tempo.

Para isso, foram criados cinco níveis de normalização. O ideal é que pelo menos a base de dados seja normalizada até o terceiro nível antes da realização de uma consulta, o que resguarda que o retorno será eficiente e que a base de dados tem nível aceitável de qualidade nos seus dados e tabelas.

Leituras recomendadas

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2010.

HEUSER, C. A. *Projeto de banco de dados*. 6. ed. Porto Alegre: Bookman, 2010. (Série Livros Didáticos UFRGS, v. 4).

KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. *Sistema de banco de dados*. 6. ed. Rio de Janeiro: Campus, 2012.

RAMAKRISHNAN, R. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: Penso, 2009.

SETZER, V. W.; SILVA, F. S. C. *Bancos de dados*. São Paulo: Blucher, 2005.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS