MODELAGEM E DESENVOLVIMENTO DE BANCO DE DADOS

Pedro Henrique Chagas de Freitas

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linguagem de definição de dados (*Data Definition Language*) (DDL)

Objetivos de aprendizagem

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

- Identificar a linguagem DDL.
- Exemplificar a linguagem DDL.
- Implementar a linguagem DDL.

Introdução

Neste capítulo, você vai estudar a modelagem e o desenvolvimento de bancos de dados a partir da perspectiva da DDL (*Data Definition Language*), observando as estruturas que compõem a DDL, bem como exemplos de sua utilização e a sua implementação na criação, alteração e exclusão de estruturas dentro de bancos de dados.

A DDL é uma linguagem de definição utilizada na composição do SQL (*Structured Query Language*), que é uma linguagem estruturada e difundida para bancos de dados relacionais e inspirada em álgebra relacional. Logo, a DDL 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 DDL

A linguagem DDL nasce nos anos 1980, dentro do contexto de migração de dados armazenados em repositórios para sistemas relacionais que deram vazão aos chamados SGBDs (Sistemas de Gerenciamento de Bancos de Dados). Quando o dado bruto começou a ser explorado, a fim de gerar informações

que, por sua vez, seriam derivadas em conhecimento, os relacionamentos entre os dados precisaram ser organizados em bancos de dados que deveriam ter alguma linguagem de criação, manipulação, transação e controle para orquestração das implementações dos projetos de bancos de dados.

A IBM começa, então, nos anos 1970, um projeto nos seus laboratórios de pesquisa para a criação de uma linguagem para bancos de dados que, com o tempo, chegou ao que hoje conhecemos como SQL (*Structured Query Language*). O SQL se tornou, então, a abstração de linguagem para banco de dados mais difundida, sendo utilizada por diversos SGBDs, como Oracle, DB2, MySQL, etc. A ideia inicial foi, então, criar uma linguagem de definição das estruturas de dados dentro do banco de dados, de modificação de dados e de especificação de segurança no acesso aos dados.

Para que isso fosse possível, o SQL precisava surgir de forma não procedural, isto é, o SQL é uma linguagem declarativa, na qual existe uma aproximação da linguagem do usuário. Na modelagem e no desenvolvimento de bancos de dados, é necessário compreender o dado nas suas diferentes perspectivas. Pensando nisso, Elmasri e Navathe (2010) definem o banco de dados como uma coleção de dados relacionados, que são entendidos como fatos que têm um significado implícito. Todavia, Korth, Silberchatz e Sudarshan (2012) ampliam esse conceito, dizendo que um banco de dados é uma coleção de dados inter-relacionados que contém informações relevantes dentro de um determinado contexto.

Logo, é possível perceber que um banco de dados representa uma coleção de dados que pode ser compreendida em relação ao número de usuário (monousuário e multiusuário), à localização dos dados (centralizado e distribuído), à utilização (operacional, DW, documental, etc.), à estrutura (estruturado, semiestrutura, não estruturado). Essa grande diversificação de bancos de dados também é responsável pela grande diversidade de SGBDs (Sistemas de Gerenciamento de Banco de Dados).

Os SGBDs são softwares que têm como foco definir, manipular e acessar bancos de dados, organizando seu funcionamento estrutural com o objetivo de evitar inconsistências, desnormalizações e redundâncias. Assim, temos conjuntos de dados (bancos de dados) sendo gerenciados por sistemas gerenciadores de bancos de dados (SGBDs). Dentre vários SGBDs, podemos destacar: MySQL, Access, PostgreSQL, Oracle 11g, etc. Para realizar de forma eficiente a gestão dos bancos de dados, os SGBDs devem manter a integridade das transações no banco de dados e a integração dos dados, minimizando inconsistências e autorizando e restringindo acessos (permissões) no banco de dados.

Para implementar, então, a gestão sobre o banco de dados, é necessário a utilização do SQL ou de outra linguagem para banco de dados; no caso, o SQL é o mais difundido e, por meio dele, as transações no banco de dados podem ser realizadas. Os bancos de dados vão apresentar três representações de abstração no momento da sua modelagem (nível físico, nível lógico e nível visão) segundo ensinam Korth, Silberschatz e Sudarshan (2012).

Dentro desse contexto, no nível físico (nível mais baixo de representação), temos a descrição de armazenamento dos dados, ou seja, temos a forma como os dados estão fisicamente armazenados. No nível lógico, temos quais dados estão armazenados e seus relacionamentos, ou seja, quais relações existem entre os dados; logo, temos, aqui, a representação da estrutura lógica dos dados. Por fim, temos o nível visão, no qual é apresentada uma abstração dos dados na forma pela qual eles são vistos pelos usuários do sistema gerenciador de banco de dados. Para operar essas três estruturas, temos a linguagem SQL, que foi dividida em quatro ramificações:

- DDL (*Data Definition Language*);
- DML (*Data Manipulation Language*);
- DTL (*Data Transaction Language*);
- DCL (*Data Control Language*).

Fique atento

Existem autores que ainda apontam que a DQL seria uma linguagem de consulta. Como ela só tem um comando (SELECT), para alguns, não é uma ramificação. Todavia, o SELECT é utilizado costumeiramente em transações em bancos de dados.

Vale relatar, ainda, dois tipos de situações comuns diante da modelagem de bancos de dados: a independência de dados física e a independência de dados lógica. A independência de dados física é a capacidade de modificar o esquema físico sem modificar o esquema lógico, enquanto a independência lógica é a capacidade de alterar o esquema lógico sem modificar esquemas externos.

Assim, o projeto de modelagem e desenvolvimento de bancos de dados vai respaldar a utilização do SQL nas transações. A DDL (*Data Definition Language*), especificamente, apresenta três instruções (Figura 1):

- CREATE (criação de tabelas);
- ALTER (modificação de tabelas);
- DROP (eliminação de tabelas).

Embora esses sejam os principais com a evolução do SQL, comandos como TRUNCATE e RENAME também foram sendo adicionados.

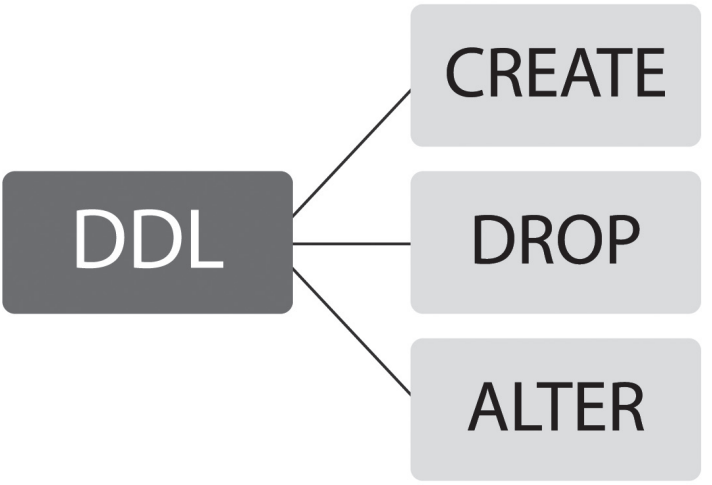

Figura 1. Instruções da DDL.

Fique atento

Os elementos DDL não são usados somente para criar, alterar ou excluir tabelas, mas também em outras estruturas, como:

- CREATE INDEX: cria um novo índice em uma tabela.
- CREATE DOMAIN: cria um tipo de dados definido pelo usuário.
- ALTER INDEX: altera um índice de uma tabela.
- DROP INDEX: exclui um índice existente em uma tabela.

Exemplificando a linguagem DDL

Temos, então, que a DDL compõe o SQL e é utilizada na modelagem e no desenvolvimento de bancos de dados para a definição de estruturas de dados, como colunas, tabelas, linhas e índices.

- CREATE TABLE (Tabelas são criadas):

```
CREATE TABLE Carro (
codChassi char(3) NOT NULL,
nomeProprietario varchar(80),
codVendedor int NOT NULL,
PRIMARY KEY (codChassi);
FOREIGN KEY (nomeProprietario) REFERENCES EMPREGADO (codVendedor)
ON DELETE SET NULL);
```

- ALTER TABLE (Tabelas são alteradas):

```
ALTER TABLE Proprietario ADD Endereco VARCHAR (30)
ALTER TABLE SistemaSolar ADD Planetas VARCHAR (10)
ALTER TABLE Predio ADD Piscinas VARCHAR (15)
```

- DROP TABLE (Tabelas são excluídas):

```
DROP TABLE Proprietario
DROP TABLE Carro
DROP TABLE Predio
```

Saiba mais

Alguns autores trazem, ainda, uma definição muito pouco utilizada, que é o VDL (*View Definition Language*). O VDL é utilizado para especificar as visões dos usuários e seus mapeamentos para o esquema lógico.

Exemplo:

CREATE VIEW (Cria uma View)

DROP VIEW (Deleta uma View)

É importante lembrar que uma VIEW é uma tabela virtual que pode ter linhas e colunas de tabelas relacionadas, sendo o resultado de uma consulta (SELECT).

Implementação da DDL

Temos, então, alguns exemplos de implementação da DDL. Se desejássemos criar uma tabela que representasse os servidores públicos de determinado Ministério, conforme mostra a Figura 2, faríamos da seguinte forma:

```
CREATE SCHEMA MINISTERIO;

CREATE TABLE SERVIDORPUBLICO (
codCPFSERVIDOR int NOT NULL,
nomeServidor varchar(80),
codCoordenador int,

PRIMARY KEY (codCPFSERVIDOR);
FOREIGN KEY (codCoordenador) REFERENCES SERVIDORPUBLICO
(codCPFSERVIDOR));

CREATE TABLE COORDENACAO (
codCoodenacao char(3) NOT NULL,
nomeCoordenacao varchar(80),
codCoordenador int NOT NULL,

PRIMARY KEY (codCoordenacao);
FOREIGN KEY (codCoordenador) REFERENCES SERVIDORPUBLICO
(codCPFSERVIDOR)
ON DELETE SET NULL);
```

Temos, então:

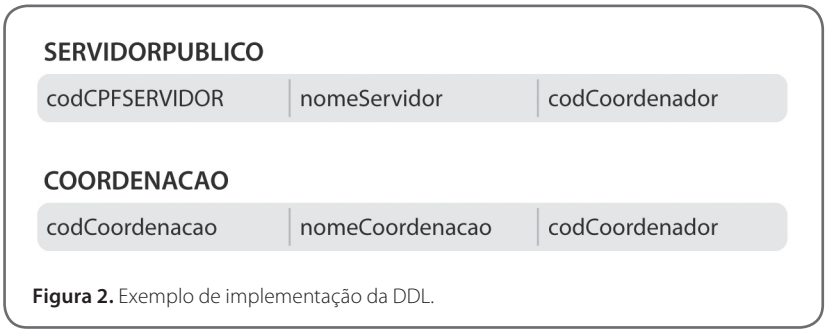

Figura 2. Exemplo de implementação da DDL.

Se o objetivo fosse implementar uma tabela "Projeto" e adicionar os nomes dos responsáveis pelos projetos, conforme mostra a Figura 3, faríamos o seguinte:

```
CREATE TABLE PROJETO (
codProjeto int NOT NULL,
nomeProjeto varchar(80),
codCoordenacaoResponsavel char(3) DEFAULT 'D01',

PRIMARY KEY (codProjeto);
FOREIGN KEY (codCoordenacaoResponsavel) REFERENCES COORDENACAO
(codCoordenacao) ON DELETE CASCADE);
```

Figura 3. Implementação da tabela "Projeto".

Para criar uma tabela "trabalha" e "administra" com CPF e o codProjeto, deve-se fazer o seguinte (Figura 4):

```
CREATE TABLE TRABALHA (
codCPFSERVIDOR int NOT NULL,
codProjeto int NOT NULL,
PRIMARY KEY (codCPFSERVIDOR, codProjeto);

FOREIGN KEY (codCPFSERVIDOR) REFERENCES SERVIDORPUBLICO
(codCPFSERVIDOR);

FOREIGN KEY (codProjeto) REFERENCES PROJETO (codProjeto);
UNIQUE (codCPFSERVIDOR);
UNIQUE (codProjeto));

CREATE TABLE ADMINISTRA (
codAdministra int NOT NULL);
```

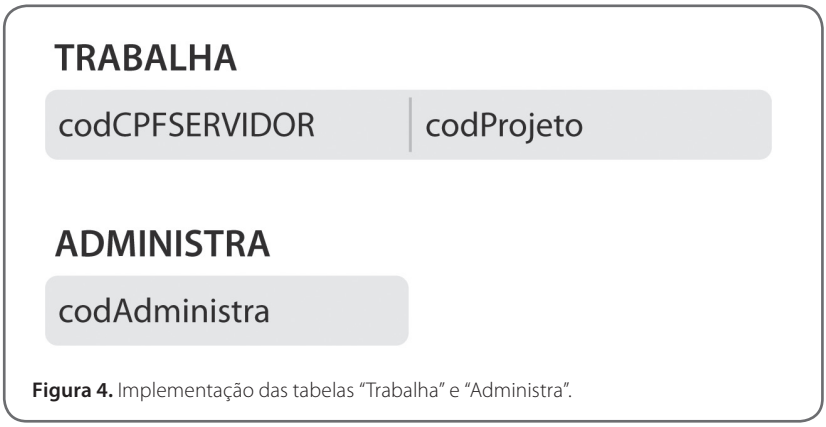

Figura 4. Implementação das tabelas "Trabalha" e "Administra".

O que fizemos afinal de contas? Nós criamos um esquema: Ministério, composto, na nossa base de dados, por cinco tabelas (SERVIDOR PUBLICO, COORDENACAO, PROJETO, TRABALHA e ADMINISTRA). Para cada tabela, definimos domínios e criamos chaves (*primary key* e *foreign key*).

Note que buscamos definir uma formação lógica e exprimi-la em comando para registro no banco de dados a partir da criação de tabelas no esquema: MINISTÉRIO. Nos Ministérios, por exemplo, temos: "Servidores Públicos", que estão vinculados a "Coordenações", que, por sua vez, possuem "Projetos"; cada servidor "Trabalha" dentro de alguma "Coordenação" em algum "Projeto" e cada "Coordenador" vai "Administrar" esses "Projetos".

O que podemos concluir aqui? Podemos concluir que os esquemas dentro de um banco de dados buscam representar a realidade do mundo exterior — no nosso caso, do Ministério. Ao tentar fazer isso, utilizamos tabelas, que são criadas a partir de comandos SQL — neste caso, CREATE TABLE. Caso desejássemos alterar essa tabela que criamos, utilizaríamos ALTER TABLE. Veja o exemplo a seguir:

```
ALTER TABLE COORDENACAO ADD Diretoria;
ALTER TABLE SERVIDORPUBLICO ADD Salario;
```

Caso desejássemos excluir uma tabela, faríamos, por exemplo:

```
DROP TABLE ADMINISTRA;
DROP TABLE ADMINISTRA CASCADE;
```

Observe que o comando "Cascade" é utilizado para excluir as referências nas quais a tabela ADMINISTRA está presente.

Referências

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

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

Leituras recomendadas

HEUSER, C. A. *Projeto de banco de dados*. 6. ed. Porto Alegre: Bookman, 2008. (Série Livros Didáticos Informática UFRGS, v. 4).

RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de bancos de dados*. 3. ed. São Paulo: McGraw-Hill, 2008.

SETZER, V. W. *Banco de dados:* conceitos, modelos, gerenciadores, projeto lógico, projeto físico. São Paulo: Blücher, 1990.

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