Banco de Dados I

Unidade 4 – SQL

QI FACULDADE & ESCOLA TÉCNICA
Curso Técnico em Informática

SUMÁRIO

INTRODUÇÃO A LINGUAGEM SQL

Nesta unidade você irá aprender sobre conceitos da linguagem SQL. São apresentados comandos básicos de criação de banco de dados e tabelas (DML).

1 SQL

SQL (*Structure Query Language*) é a Linguagem de Consulta Estruturada que surgiu dentro dos laboratórios de pesquisa da IBM na década de 70.

A linguagem SQL foi declarada um padrão pelo ANSI (*American National Standarts Institute*) e pela ISO (*International Organization for Standardizion)* com diversos comandos padrões de DDL (Linguagem de Definição de Dados) e DML (Linguagem de Manipulação de Dados) lançados desde seu desenvolvimento inicial.

O SQL Padrão ANSI é o modelo padrão de comandos SQL que define objetos básicos de banco de dados como tabelas, visões e índices, mas cada banco de dados cria e evolui os padrões com comandos próprios, portanto existem diferenças entre a sintaxe usada para acessar cada SGBD.

O conceito de SQL basicamente está dividido em duas partes:

- ✓ Comandos que definem e manipulam a estrutura de armazenamento e procedimentos;
- ✓ Comandos que manipulam os dados.

Os profissionais de banco de dados se referem as operações básicas de banco de dados como CRUD:

- ✓ **C** – *Create* – criar ou adicionar dados;
- ✓ **R** – *Read* – ler dados;
- ✓ **U** – *Update* – atualizar os dados;
- ✓ **D** – *Delete* – excluir dados.

1.1 MYSQL

Banco de dados relacional utilizado para sistemas de internet. Criado por David Axmark, Allan Larsson e Michael "Monty" Widenius em 1998. Foi adquirido pela Sun MicroSystem. Em 2009 a Oracle comprou a Sun e passou a ter a propriedade do MySQL.

2 COMANDOS SQL

Os comandos SQL são divididos em:

- ✓ **Data Definition Language (DDL)** - Linguagem de definição de dados, utilizada para criar e manter as estruturas de armazenamento usadas no banco de dados.
- ✓ **Data Manipulation Language (DML)** - Linguagem de manipulação de dados, utilizada para incluir, alterar, excluir e consultar dados nas estruturas do banco de dados.
- ✓ **Data Control Language (DCL)** – Linguagem de controle de dados de um banco de dados e do controle de usuários do banco de dados;
- ✓ **Data Query Language (DQL):** Linguagem de Consulta de Dados;
- ✓ **Data Transaction Language (DTL)**: Linguagem de Transação de Dados.

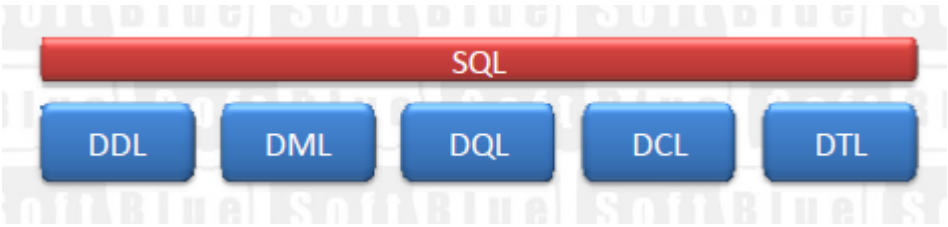

Figura 1 – Estrutura dos Comandos SQL
Fonte: Softblue

3 DDL

DDL (***Data Definition Language***) é a linguagem de definição de dados, utilizada para criar e manter as estruturas de armazenamento usadas no banco de dados. Os comandos DDL não acessam os dados, mas interferem em sua existência ou forma de armazenamento e acesso.

Exemplo de Comandos:

- ✓ ***Create***: Cria uma estrutura: banco de dados, tabelas e outros;
- ✓ ***Alter***: Altera uma estrutura: banco de dados, tabelas e outros;
- ✓ ***Drop***: Exclui uma estrutura: banco de dados, tabelas e outros.

3.1 Criação do Banco de Dados

O comando **CREATE DATABASE** serve para a criação de um novo banco de dados. No MySQL os bancos de dados são implementados como diretórios contendo arquivos que correspondem a tabelas do banco de dados.

Sintaxe:

```
CREATE DATABASE <nome_banco>;
```

Exemplo: Criar um banco de dados para armazenar as informações de uma biblioteca:

```
CREATE DATABASE Biblioteca;
```

Após criar o banco, indicamos que iremos trabalhar nele com o comando USE:

```
USE Biblioteca;
```

3.2 Criação de Tabela

Comando **CREATE TABLE** serve para criar uma tabela fisicamente no banco de dados. Pequenas variações podem ocorrer de acordo com o banco de dados.

Sintaxe:

```
CREATE TABLE <nome_tabela> (<campos> <opções>);
```

Exemplo: No sistema de bibliotecas criar a tabela dos gêneros dos livros.

```
CREATE TABLE genero (
    codigo_genero int,
    nome varchar(30)
);
```

3.2.1 Tipos de Dados no MySQL

Os tipos de dados são uma forma de classificar as informações que serão armazenados no banco de dados. Conforme o banco de dados que utilizamos, a classificação dos tipos de dados e seus comportamentos podem variar. A escolha do tipo de dado para cada campo de uma tabela deve ser feito com muita atenção e cuidado.

Os tipos de dados dos campos das tabelas segundo o MySQL são ilustrados nas tabelas a seguir.

Tabela 1 - Tipos de dados numéricos

Tipo de Dado	Uso	Observação	Exemplo declaração
INT or INTEGER	Um inteiro de tamanho normal	Pode ser Unsigned	Ano de nascimento: Ano int unsigned
FLOAT Floating-Point Types (Valores aproximados)	Um pequeno número de ponto flutuante (precisão simples). Na declaração indicar tamanho e quantidade de casas decimais.	Não pode ser unsigned	Distância em metros: Distancia float(7,2)
DOUBLE, DOUBLE PRECISION, REAL Floating-Point Types (Valores aproximados)	Um número de ponto flutuante de tamanho normal (precisão dupla)	Não pode ser unsigned	Distância em metros: Distancia double(10,4)
DECIMAL, NUMERIC *Fixed-Point Types* *(Valores exatos)*	Um número de ponto flutuante exato, bastante indicado para armazenar valores monetários onde a precisão é muito importante.	Não pode ser unsigned	Salário do funcionário Salario decimal(5,2)

Tabela 2 - Tipos de dados para caracteres

Tipo de Dado	Uso	Observação	Exemplo declaração
CHAR	String de tamanho fixo. Sempre é completada com espaços a direita até o tamanho definido.	De 1 a 255 caracteres	RG de uma pessoa rg char(10)
VARCHAR	String de tamanho variável	De 1 a 255 caracteres	Nome varchar(200)
TEXT	Textos	65535 caracteres	Um campo para armazenar o texto de uma notícia Texto text

Tabela 3 - Tipos de dados para data e hora

Tipo de Dado	Uso	Formato	Exemplo declaração
DATE	Para armazenar datas	Formato: 'YYYY-MM-DD'	Nascimento date
DATETIME	Para armazenar data e hora no mesmo campo	'YYYY-MM-DD HH:MM:SS'	Datahora_compra datatime
TIME	Para armazenar horas	'HH:MM:SS'	Hora_entrada time

***Dica de leitura**: saiba mais sobre os tipos de dados do MySQL em: http://imasters.com.br/artigo/9196/mysql/tipos-de-dados-do-mysql-50*

3.2.2 Atributos dos campos

Vejamos abaixo alguns dos atributos dos campos que são usados na criação ou alteração dos dados:

- ✓ **NULL**: Permite que o campo aceite valores nulos (vazio). Pode ser usado para diferenciar entre um valor zerado e um campo que nunca tenha tido valor.
- ✓ **NOT NULL**: Não permite valores nulos.
- ✓ **AUTO_INCREMENT**: criar um código sequencial que é gerado automaticamente pelo banco de dados.
- ✓ **UNIQUE**: Não permite duplicidade de valor, ou seja, requer valores únicos. Aceita somente um registro com conteúdo nulo.
- ✓ **DEFAULT:** Atribui um valor padrão ao campo, quando um novo registro for criado sem especificar o valor dele.
- ✓ **SIGNED/UNSIGNED:** para tipos de dados numéricos. Determina se o número terá ou não sinal (-). *Unsigned* indicará que o campo só aceitará números positivos.

3.2.3 Chave Primária (PK)

A Chave Primária é o atributo identificador sendo uma chave única, universal e imutável. Definição de uma chave primária (**PRIMARY KEY**) pode ser na criação da tabela ou depois quando a tabela já existe.

Na criação da tabela use a sintaxe:

```
CREATE TABLE <nome_tabela> (
    <campo> <tipo> NOT NULL PRIMARY KEY
);
```

Ou

```
CREATE TABLE <nome_tabela> (
   <campo> <tipo> NOT NULL,
   PRIMARY KEY <campo>
);
```

Na alteração de uma tabela para incluir a chave primária use a sintaxe:

```
ALTER TABLE <nome_tabela> ADD PRIMARY KEY (<campo>);
```

*Para usar a sintaxe acima, o campo deve estar definido como **NOT NULL** e deve estar preenchido em todos os registros da tabela e não ter registros duplicados.*

Exemplo: no sistema de bibliotecas criar a tabela dos gêneros dos livros. O código do gênero será numérico, não nulo, gerado automaticamente pelo sistema e chave primária. E o nome será alfanumérico com até 30 caracteres, não pode ser nulo.

```
CREATE TABLE genero (
  codigo_genero int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nome varchar(30) NOT NULL
);
```

3.2.4 Chave Primária Composta

Ocorre quando uma tabela tem mais de um atributo como chave primária. Composição de duas ou mais colunas para gerar uma combinação única.

Exemplo: Um aluno paga todo mês a mensalidade do curso e essa mensalidade está vinculada a somente ele. Então na mensalidade temos 2 chaves: matrícula do aluno e o mês/ano.

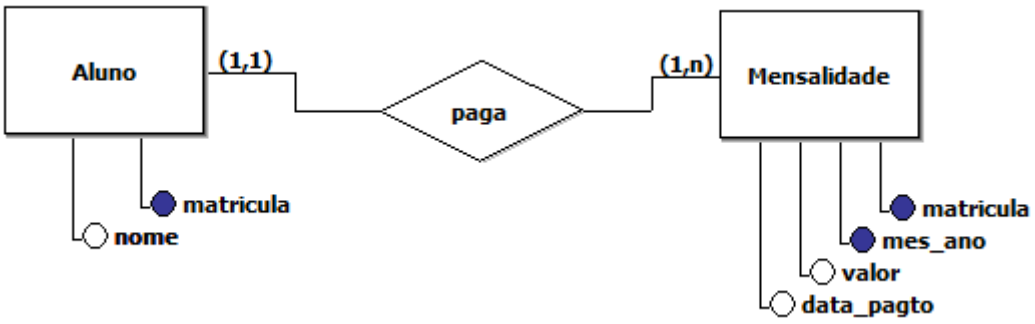

Figura 3 – Chave Primária Composta

No modelo relacional representamos com os dois campos sublinhados: Mensalidade(matricula, mes_ano, valor, data_pagto).

A seguir vemos o exemplo de como ficariam os dados inseridos na tabela "mensalidade" do banco de dados.

Matricula	Mês_ano	Valor	Data_pagto
0010	062015	300,00	10/06/2015
0010	072015	300,00	10/07/2015
0010	082015	300,00	10/08/2015
0010	092015	300,00	10/09/2015
0010	102015	300,00	10/10/2015
0010	112015	310,00	10/11/2015
0010	122015	310,95	10/12/2015

Figura 4 – Tabela Mensalidade

A seguir o comando SQL para criar a tabela mensalidade com chave primária composta campos "matricula" e "mesano".

```
CREATE TABLE mensalidade (
   matricula varchar(8) NOT NULL,
   mesano varchar(6) NOT NULL,
   valor double NOT NULL,
   data_pagto date NOT NULL,
   PRIMARY KEY (matricula, mesano)
);
```

3.2.5 Chave Estrangeira (FK)

A chave estrangeira é um atributo que se relaciona com outra tabela, ocorre quando um atributo de uma relação for chave primária em outra relação. Sempre que houver o relacionamento 1:N entre duas tabelas, a tabela 1 receberá a chave primária e a tabela N receberá a chave estrangeira.

Uma chave estrangeira é sempre baseada em um valor único da entidade referenciada, normalmente a chave primária. A definição de uma chave estrangeira (***FOREIGN KEY***) pode ser feita na criação ou na alteração da tabela.

Na criação da tabela use a sintaxe:

```
CREATE TABLE <nome_tabela> (
   <campo> <tipo>,
   FOREIGN KEY (<campo>) REFERENCES <outra_tabela>
(<campo_outratabela>));
```

Na alteração da tabela use a sintaxe:

```
ALTER TABLE <nome_tabela> ADD FOREIGN KEY (<campo>) REFERENCES
<outra_tabela> (<campo_outratabela>);
```

Exemplo: No sistema da biblioteca a tabela livro possui referência às tabelas gênero, editora e autor.

```
CREATE TABLE Livro (
   codigo_livro int PRIMARY KEY,
   nome varchar(50) NOT NULL,
   ano int,
   edicao int NULL,
   codigo_autor int,
   codigo_genero int,
   codigo_editora int,
   FOREIGN KEY (codigo_autor) REFERENCES autor (codigo_autor),
   FOREIGN KEY (codigo_genero) REFERENCES genero (codigo_genero),
   FOREIGN KEY (codigo_editora) REFERENCES genero (codigo_editora)
);
```

3.2.6 Chave Candidata

Uma chave candidata ou alternativa ocorre quando em uma relação existe mais de uma combinação de atributos possuindo a propriedade de identificação única. A chave candidata é apenas conceitual. Os atributos com essas características poderiam ser chave primária já que possuem por natureza a identificação única.

Exemplos: CNH, CPF, CNPJ, RG, Título Eleitor, entre outros.

3.3 Atualizar a estrutura de uma tabela

O comando **ALTER TABLE** altera a estrutura de uma tabela existente fisicamente dentro do banco de dados. As regras de negócio não podem ser violadas. São três as formas de atualização: adicionar, modificar e excluir.

Sintaxe:

```
ALTER TABLE <nome_tabela> <ação> <campos>;
```

3.3.1 Adicionar uma coluna

Para adicionar uma coluna usa-se o comando **ALTER TABLE** ... **ADD**.

Sintaxe:

```
ALTER TABLE <nome_tabela> ADD (<campo>)
```

Exemplo: No sistema de biblioteca adicionar o campo sigla na tabela de gênero dos livros.

```
ALTER TABLE genero ADD sigla VARCHAR(3) NULL;
```

3.3.2 Modificar uma coluna já existente

Para modificar uma coluna usa-se o comando **ALTER TABLE** ... **CHANGE**. No Oracle usa-se o comando **ALTER TABLE** ... **MODIFY**.

Sintaxe:

```
ALTER TABLE <nome_tabela> CHANGE (<campo> <tipo_campo> <opções>)
```

Exemplo: No sistema de biblioteca alterar o campo código para codigo_editora e auto incremento da tabela editora.

```
ALTER TABLE editora CHANGE codigo codigo_editora INT(5) NOT NULL AUTO_INCREMENT;
```

3.3.3 Excluir uma coluna de uma tabela

Para excluir uma coluna de uma tabela usa-se o comando **ALTER TABLE ... DROP**.

Sintaxe: **ALTER TABLE** <nome_tabela> **DROP** (<campo>)

Exemplo: No sistema da biblioteca excluir o campo teste da tabela autor dos livros.

```
ALTER TABLE autor DROP teste;
```

3.4 Excluir uma tabela ou banco de dados

Excluir o banco de dados ou uma tabela existente fisicamente do banco de dados.

Sintaxe:

```
DROP TABLE <nome_tabela>;
```

Excluir um banco de dados.

```
DROP DATABASE <<nome_banco>;
```

Exemplo: No sistema de biblioteca excluir a tabela livro do banco de dados:

***DROP TABLE** livro;*

Os relacionamentos da tabela a ser deletada devem ser desativados antes da exclusão senão o comando poderá não funcionar.

4 EXEMPLO DO BANCO DE DADOS

No sistema de biblioteca são cadastrados os livros que são emprestados para os alunos. Os livros possuem somente um autor principal cadastrado para fins de estudo de caso, gênero do livro e a editora que o publicou.

A figura ilustra o modelo conceitual da biblioteca.

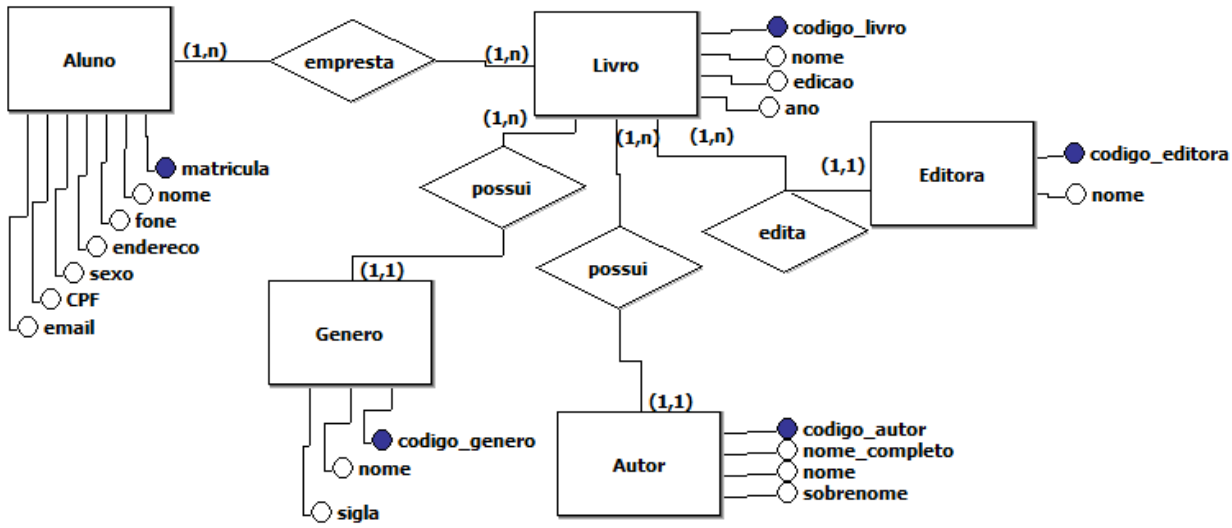

Figura 5 – Modelo Conceitual Biblioteca

Com base no modelo conceitual foi criado o modelo relacional a seguir.

Tabela que armazena os dados dos alunos:

Aluno(<u>matricula</u>, nome, fone, endereço, sexo, CPF, email).

Tabela que armazena as editoras dos livros:

Editora(<u>codigo_editora</u>, nome)

Tabela que armazena os autores dos livros:

Autor(codigo_autor, nome_completo, nome, sobrenome)

Tabela que armazena os gêneros dos livros:

```
Genero(codigo_genero, nome, sigla)
```

Tabela que armazena os dados dos livros:

```
Livro(codigo_livro, nome, edição, ano, codigo_autor, codigo_genero, codigo_editora)
codigo_autor referencia Autor (codigo_autor)
codigo_genero referencia Genero (codigo_genero)
codigo_editora referencia Editora (codigo_editora)
```

A tabela que armazena o relacionamento N:N entre os alunos que emprestam vários livros:

```
Emprestimo(codigo_emprestimo, matricula, codigo_livro, data_retirada, data_devolucao)
matricula referencia Aluno (matricula)
codigo_livro referencia Livro (codigo_livro)
```

A seguir o banco de dados criado no MYSQL para o estudo de caso da biblioteca.

Criação do banco de dados da biblioteca:

```
CREATE DATABASE Biblioteca;
USE Biblioteca;
```

Criação da tabela de gênero dos livros:

```
CREATE TABLE genero (
  codigo_genero int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nome varchar(30) NOT NULL
);
```

Criação da tabela da editora dos livros:

```
CREATE TABLE editora (
  codigo_editora int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nome varchar(50) NOT NULL
);
```

Criação da tabela livro:

```
CREATE TABLE Livro (
  codigo_livro int PRIMARY KEY,
  nome varchar(50) NOT NULL,
  ano int,
  edicao int,
  codigo_autor int,
```

```
  codigo_genero int,
  codigo_editora int,
  FOREIGN KEY (codigo_autor) REFERENCES autor (codigo_autor)
  FOREIGN KEY (codigo_genero) REFERENCES genero (codigo_genero),
  FOREIGN KEY (codigo_editora) REFERENCES genero (codigo_editora)
);
```

Criação da tabela aluno:

```
CREATE TABLE aluno (
  matricula varchar(8) NOT NULL PRIMARY KEY ,
  nome varchar(50) NOT NULL,
  fone varchar(15),
  endereco varchar(100),
  sexo varchar(1),
  CPF varchar(11) NOT NULL,
  email varchar(50)
);
```

Criação da tabela empréstimo dos livros:

```
CREATE TABLE Emprestimo (
  codigo_emprestimo INT NOT NULL,
  matricula varchar(8) NOT NULL,
  codigo_livro int NOT NULL,
  data_retirada date,
  data_devolucao date,
  PRIMARY KEY(codigo_emprestimo,matricula,codigo_livro),
  FOREIGN KEY(matricula) REFERENCES Aluno (matricula),
  FOREIGN KEY(codigo_livro) REFERENCES Livro (codigo_livro)
);
```

REFERÊNCIAS

GILLENSON, Mark L. et al. **Introdução à Gerência de Banco de Dados**. Rio de Janeiro: LTC, 2009.

HEUSER, Carlos Alberto. **Projeto de Banco de Dados**. Porto Alegre: Bookman, 2009.

NISHIMURA, Roberto Yukio. **Banco de Dados I**. São Paulo: Person Prentice Hall, 2009.

PRATES, Rubens. **MYSQL – Guia de Consulta Rápida**. São Paulo: Novatec, 2000.

SOFTBLUE. **Curso SQL Completo**. Disponível via web em http://www.softblue.com.br/. Acessado em 2015.

TAKAHASHI, Maná. **Guia Mangá de Banco de Dados**. São Paulo: Novatec, 2009.