SQL (Structured Query Language) é a linguagem padrão em sistemas de gerenciadores de bancos de dados relacionais

Trata-se de uma linguagem de consulta e de grande importância para desenvolvedores, pois é com ela que podemos nos comunicar com os bancos de dados utilizados pelas aplicações.

Como utilizar o SQL?

se trata de uma linguagem declarativa

DDL - Data Definition Language - Conjunto de comandos que lidam com os objetos, criando bancos de dados, esquemas, tabelas, campos, etc. Dentre os mais utilizados temos

CREATE: CREATE TABLE produto -- Criação da tabela produto

ALTER:

DROP:

DML - Data Manipulation Language - Os comandos aqui lidam com os dados. Alguns muito comuns são

INSERT: INSERT INTO produto (nome, valor) VALUES (‘geladeira’,’1200’)

-- Inserção de valores na tabela produto

UPDATE:

DELETE:

DQL - Data Query Language - Linguagem de consulta de dados conta com o conjunto da instrução utilizada para a obtenção dos registros dos bancos de dados. Exemplo:

SELECT nome, valor FROM produto -- Recupera valores da tabela produto

O XAMPP é um servidor independente de sistema operacional, onde o mesmo contém o MySQL, porém atualmente foi substituído pelo MariaBD, um servidor web Apache e os interpretadores para as linguagens PHP e Perl. Uma das vantagens de utilizarmos o XAMP para instalar o MySQL, por exemplo, é a facilidade na instalação, pois rapidamente temos um SGBD pronto para ser utilizado.

CREATE DATABASE bd\_devmedia DEFAULT CHARACTER SET utf8;

Nesse script temos a criação do banco de dados onde foi definido o CHARACTER SET utf8, pois na hora de escolhermos o Collation, definimos que seria um dos padrões usados pelo utf8, logo, não podemos ter um CHARACTER SET diferente de um Collation.

Configurações dos campos

[Column Name]: Define o nome de uma coluna na tabela, existem regras para criarmos campos, como por exemplo, não utilizar acentos e espaços, mantendo assim o padrão da linguagem SQL.

[DataType]: O datatype define o tipo do campo que estamos criando, por exemplo, varchar é um tipo de campo que aceita qualquer caractere, já o int, só aceitará números inteiros.

[Chave primária]: A chave primária define a identificação de um registro exclusivo e único, por exemplo, se a chave primária da tabela de clientes for IDCliente, nunca teremos mais de um cliente com o ID = 1, por exemplo. Outra função da chave primária é criar relacionamento com as demais tabelas.

[Not Null]: O not null define que um determinado campo precisa ter algum tipo de informação, por exemplo, se o nome for NN, nunca poderemos criar um registro na tabela sem que este campo seja preenchido.

[Unique Index]: O unique index garante que determinado campo na sua tabela não será repetido, porém não podemos criar relacionamentos entre tabelas usando esse tipo.

[Auto incremento]: O auto incremente geralmente é utilizado juntamente com a chave primária, para garantir que a mesma não se repita. O próprio SGBD se encarrega de incrementar +1 no campo que está configurado como AI, por exemplo, ao incluir um novo registro na tabela.

[Zero-Filled]: O flag ZF é utilizado quando queremos incluir um determinado número de zeros a esquerda de um campo do tipo INT, por exemplo. Imagine que tenhamos um campo INT com tamanho 5, o flag ZF irá preencher o campo com zeros a esquerda até completar cinco caracteres. Se informarmos 10, o que será armazenado no campo será 00010, e não apenas 10, caso não marquemos o ZF, geralmente utilizamos ZF para campos de matrícula ou códigos de barras, por exemplo.

[Default/Expression]: O Default/Expression tem a finalidade de definirmos uma valor padrão ou uma expressão para um determinado campo, por exemplo, um cadastro de clientes onde a maior parte dos clientes do sexo masculino, podemos definir no campo sexo o valor default como ‘M’, evitando assim ter que preencher o sexo toda hora que inserirmos um registro.

Criando tabela

CREATE TABLE `cliente` (

`idCliente` int(11) NOT NULL AUTO\_INCREMENT,

`Nome` varchar(100) NOT NULL,

`Sexo` char(1) DEFAULT 'M',

`DataNascimento` date DEFAULT NULL,

`Peso` double DEFAULT NULL,

PRIMARY KEY (`idCliente`),

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

Inserindo dados

Após criar a nossa tabela o próximo passo é incluir um registro. Abaixo veremos o script para incluir um registro na tabela de cliente.

insert into cliente

(

nome

, sexo

, datanascimento

, peso

)

values ('Wesley', 'M', '1987-10-05', 91.40);

Documentação

Ao criarmos uma chave primária, automaticamente o MySQL cria um índice para gerenciar o campo definido como chave primária, porém podemos criar um index em qualquer outro campo, como, por exemplo, o campo Nome.

Imagine que sua aplicação tenha uma tela de filtro por nome do cliente, é recomendado criar um index para esse campo, facilitando assim as consultas a serem executadas.

Os índices podem ser compostos, ou seja, mais de um campo podem compor um mesmo index, como é o caso abaixo, onde temos SEXO+PESO formando um index. A instrução para criar um index composto é mostrada abaixo:

1. ALTER TABLE ‘bd\_devmedia’.’cliente’

2. ADD INDEX 'Index\_Sexo\_Peso’ (‘Sexo’ ASC, ’Peso’ ASC);

Explicando o código

Linha 1:

A instrução ALTER TABLE define que a tabela cliente, do schema bd\_devmedia sofrerá alguma alteração.

Linha 2:

A instrução ADD INDEX criar um index na tabela cliente com o nome Index\_Sexo\_Peso e os campos Sexo e Peso serão indexados de forma ascendente.

Documentação

A chave estrangeira será criada no momento que criarmos a tabela telefonecliente. Essa tabela é responsável por armazenar os telefones dos clientes cadastrados em nosso banco de dados e ela possui um relacionamento 1:N, ou seja, um cliente pode ter 1 ou vários telefones e um telefone só pertence a um cliente. Abaixo veremos o script de criação da tabela de telefonecliente.

CREATE TABLE `telefonecliente` (

`idtelefonecliente` int(11) NOT NULL AUTO\_INCREMENT,

`ddd` varchar(3) NOT NULL,

`numero` varchar(10) NOT NULL,

`idcliente` int(11) NOT NULL,

PRIMARY KEY (`idtelefonecliente`),

KEY `FK\_Cliente\_idx` (`idcliente`),

CONSTRAINT `FK\_Cliente` FOREIGN KEY (`idcliente`)

REFERENCES `cliente` (`idCliente`) ON DELETE NO ACTION

ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

Explicando o código

Linha 1:

O comando CREATE table define que estamos criando uma tabela chamada telefonecliente.

Linha 2 a 5:

Define o nome dos campos, seus tipos e suas configurações adicionais: not null e auto\_increment, por exemplo.

Linha 6:

Define o campo idcliente como sendo a chave primária da tabela cliente.

Linha 7:

Define que o campo idcliente será uma chave estrangeira FK\_Cliente\_idx.

Linha 8:

Define a configuração da FK idcliente, vinculando a tabela cliente e restringindo as ações de delete e update. Caso exista uma mudança (atualização ou remoção) na chave primeira idcliente da tabela de cliente, essa restrição impedirá se houverem registros na tabela de telefonecliente vinculados ao idcliente modificado.

Linha 9:

Define a Engine, o valor inicial do auto\_increment e também o charset da tabela de cliente.

Inserindo dados em cliente

Antes de incluirmos alguns telefones, vamos inserir mais três clientes no nosso banco de dados seguindo a mesma estrutura vista em vídeos anteriores.

insert into cliente

(

nome

, sexo

, datanascimento

, peso

)

values ('Fernando', 'M', '2000-01-31', 81.40),

('Daniella', 'F', '1988-10-15', 70.20),

(‘Lia', 'F', '1984-01-05', 60.30);

Explicando o código

Linha 8 a 10:

Utilizando , depois do primeiro values podemos incluir diversas linhas na nossa tabela e o último registro a ser incluído colocamos apenas ;, opcionalmente.

Inserindo dados em telefonecliente

Após criar a nossa tabela o próximo passo é incluir vários telefones para os clientes cadastrados. Abaixo veremos o script para incluir vários registros na tabela de telefonecliente.

insert into telefonecliente

(

ddd

, numero

, idcliente

)

values ('021', '33331111', 1),

('021', '22222222', 2),

('021', '78789898', 3),

('021', '34212121', 2),

('021', '36353635', 2),

('021', '37373636', 4);

Explicando o código

Linha 1:

O comando INSERT into prepara a tabela para receber novos registros.

Linha 2 a 6:

Os campos são separados por , e a ordem que são informado é importante, pois mais a frente passaremos as informações sequencialmente para cada um destes.

Linha 7 a 12:

A instrução values define os valores que serão incluídos na nossa tabela, a sua ordem precisa ser a mesma que a instrução anterior, desta forma cada informação será incluída na respectiva coluna.