INSTITUTO FEDERAL
Rondônia

Banco de Dados I

Vilhena

Conceitos

Para a criação de banco de dados, tabelas e atributos em um SGBD, utilizaremos a linguagem SQL que é compostas de comandos de manipulação, definição, transação, seleção e controle de dados.

Comandos DDL (Data Definition Language): Conjunto de comandos responsáveis pela criação, alteração e deleção da estrutura das tabelas e índices de um sistema (DATE, 2004).

A Linguagem de definição de dados disponibiliza um conjunto de comandos para criação(CREATE), alteração(ALTER) e remoção (DROP) de tabelas e outras estruturas

Comando CREATE DATABASE

A maioria dos SGBDs disponibiliza ferramentas que permitem a criação de Bancos de Dados, mas é possível criar o próprio Banco de Dados a partir de um comando SQL.

A sintaxe do comando é:

```
CREATE DATABASE nome_banco_de_dados;
```

Comando DROP DATABASE

O comando DROP DATABASE permite remover um determinado Banco de Dados, apagando todas as tabelas e estruturas associadas e, consequentemente, todos os dados existentes nelas

A sintaxe do comando é:

```
DROP DATABASE nome_banco_de_dados;
```

Tipo de Dados

Para criar as tabelas no Banco de dados, primeiro devemos saber quais são os tipos de dados mais comuns:

Em SQL os tipos de dados são agrupados em 3 categorias:

- Caracteres (Strings)
- Numéricos
- Tempo e Data

Tipo de Dados

Para dados do tipo	Tipo definido	Tamanho
Caracteres	Char(n), varchar (n)	Armazena até n bytes
Numérico exato	Decimal (p,e) ou numeric (p, e)	Depende
Número aproximado	Float, real	8 bytes e 4 bytes
Número inteiro	Int, smallint, tinyint	4 bytes, 2 bytes e 1 byte
Data e hora	Date, Time, smalldatetime, timestamp	8 bytes, 4 bytes
Texto e imagens	Text, image, ntext	Variável
Monetário	Money, smallmoney	8 bytes, 4 bytes

Comando CREATE TABLE

O comando CREATE TABLE é o principal comando DDL da linguagem SQL. A criação de tabelas é realizada em SQL utilizando este comando. Sua sintaxe básica é a seguinte:

```
CREATE TABLE nome_da_tabela (Coluna1 Tipo,Coluna2 Tipo, ColunaN Tipo)
```

Comando CREATE TABLE

```
CREATE TABLE empregado (empr_id INTEGER, empr_nome VARCHAR(50),
emp_dt_nasc DATE, emp_salario FLOAT);
```

empr_id	empr_nome	emp_dt_nasc	emp_salario

Colunas

Para execução do comando CREATE TABLE é necessário indicar qual o nome da tabela e, para cada uma das colunas, o nome da coluna e o tipo de dados. No entanto, podem ser indicadas as características próprias de cada uma das colunas, tais como:

- Que valores pode admitir ?
- Qual o valor padrão?
- O campo representa um atributo identificador(chave primária)?

Atributo NOT NULL

```
CREATE TABLE disciplina (disc_id NUMERIC, disc_nome VARCHAR(45));
```

Estamos admitindo que a tabela é composta por duas colunas(disc_id e disc_nome) e que qualquer uma delas pode admitir valores nulos(ou seja, o usuário poderá informar dados vazios para os campos).

Se quisermos que uma coluna não admita valores nulos, usamos a cláusula NOT NULL.

```
CREATE TABLE disciplina (disc_id NUMERIC NOT NULL, disc_nome VARCHAR(45));
```

O código acima está enfatizando que o atributo disc_id da tabela disciplina, NÃO ACEITA valores nulos, ou seja, o campo disc_id deverá possuir SEMPRE, qualquer valor diferente de vazio.

Valores por padrão (DEFAULT)

Caso um valor não seja inserido em uma coluna o valor padrão (default) armazenado nela é NULL. No entanto, é possível associar um outro valor default através da cláusula DEFAULT.

Se quisermos por exemplo que a disciplina padrão (default) se chame Matemática, então teremos que fazer o seguinte:

```
CREATE TABLE disciplina (disc_id NUMERIC NOT NULL, disc_nome VARCHAR(45) DEFAULT 'Matemática');
```

Constraints (Restrições)

Restrições são regras a que os valores de uma ou mais colunas devem obedecer. A utilização de restrições é a única garantia que temos de que os dados existentes nas colunas estão de acordo com as regras especificadas no projeto do Banco de Dados.

Existem alguns tipos distintos de restrições que se podem aplicar a colunas:

- Constraint NOT NULL
- Constraint UNIQUE
- Constraint INDEX
- Constraint PRIMARY KEY
- Constraint FOREIGN KEY

Constraint UNIQUE

A constraint UNIQUE indica que os valores dessa coluna não podem se repetir. Em uma tabela podem existir tantas colunas UNIQUE quantas forem necessárias.

Veja o exemplo:

```
CREATE TABLE funcionario (
func_codigo decimal(10,0) NOT NULL,
func_nome varchar(60) NOT NULL,
func_cpf char(15) NOT NULL,
func_tmp_servico int(11) NOT NULL,
UNIQUE KEY uk_funcionario_nome (func_nome),
UNIQUE KEY uk_funcionario_cpf (func_cpf));
```

Constraint INDEX

A constraint INDEX é usado para criar e recuperar dados da tabela da forma mais eficiente possível.

Veja o exemplo:

```
CREATE INDEX ix_nome_tabela_nn ON
table_nome (column1,column2,...);
```

```
CREATE INDEX ix_cpf_func_nn ON
funcionario (func_cpf);
```

Constraint PRIMARY KEY

Define qual atributo será a chave primária da tabela. Caso ela tenha mais de uma coluna como chave, elas deverão ser relacionadas entre os parênteses. Os atributos devem ser definidos como NOT NULL

Veja o exemplo:

```
PRIMARY KEY (atributo1, atributo2, ..., atributoX)
```

Constraint FOREIGN KEY

FOREIGN KEY (nome-coluna-chave-estrangeira) REFERENCES (nome-tabela origem): serve para definição de chaves estrangeiras, ou seja, os campos que são chaves primárias de outras tabelas. Na opção REFERENCES devemos escrever o nome da tabela de onde veio a chave estrangeira, ou seja, a tabela de origem onde o campo era a chave primária (HEUSER,2004).

Constraint FOREIGN KEY

ON DELETE – caso haja exclusão de um registro na tabela de origem e existe um registro correspondente nas tabelas filhas essa opção diz os procedimentos que devem ser feitos pelo SGBD (SILBERSCHATZ e SUDARSHAN, 2012).

As opções disponíveis são:

- RESTRICT - Opção default (padrão). Não permite a exclusão na tabela de pai (tabela de origem) de um registro, cuja chave primária exista em alguma tabela filha.
- CASCADE – Realiza a exclusão em todas as tabelas filhas que possuam o valor da chave que será excluída na tabela pai.
- SET NULL - Atribui o valor NULO nas colunas das tabelas filha que contenham o valor da chave que será excluída na tabela pai.

Constraints

```
CREATE TABLE funcionario (
func_codigo decimal(10,0) NOT NULL,
func_nome varchar(60),
func_cpf char(15),
func_tmp_servico int(11),
carg_id int NOT NULL,
CONSTRAINT pk_funcionario
PRIMARY KEY (func_codigo),
CONSTRAINT fk_funcionario_cargo
FOREIGN KEY (carg_id) REFERENCES cargo (cargo_id)
ON DELETE RESTRICT,
UNIQUE KEY uk_funcionario_nome (func_nome),
UNIQUE KEY uk_funcionario_cpf (func_cpf));
```

Comando ALTER TABLE

Outro comando da linguagem DDL é o comando ALTER. Com esse comando podemos alterar a estrutura de uma tabela, por exemplo, adicionar mais atributos a uma tabela (Entidade). Essa clausula permite acrescentar modificar e alterar nomes e formatos de colunas de tabelas.

Sintaxe:

```
ALTER TABLE nome_tabela ADD nome_coluna tipo_de_dado;
```

```
ALTER TABLE nome_tabela MODIFY nome_coluna tipo_de_dado ;
```

Comando ALTER TABLE

Onde cada campo representa:

- **nome-tabela** - nome da tabela que será atualizada.
- **nome-coluna** - nome da coluna que será criada.
- **tipo-do-dado** - tipo e tamanho dos campos definidos para a tabela.
- ADD - inclusão de uma nova coluna na estrutura da tabela. Na coluna correspondente a esse campo já existente será preenchido o valor NULL (Nulo).
- MODIFY- Permite a alteração na característica da coluna especificada:

Exemplos - Comando ALTER TABLE

adicionar o atributo

```
ALTER TABLE funcionario ADD endereco VARCHAR (20);
```

modificar o atributo

```
ALTER TABLE funcionario MODIFY endereco VARCHAR (100) NOT NULL;
```

renomear o atributo

```
ALTER TABLE funcionario CHANGE endereco func_endereco VARCHAR (100) NOT NULL;
```

remover o atributo

```
ALTER TABLE funcionario DROP func_endereco;
```

Comando DROP TABLE

O comando DROP TABLE permite remover uma determinada tabela de um Banco de Dados, e consequentemente, todos os dados existentes nela.

A sintaxe do comando é:

Referências

DATE, C. J. Introdução a Sistemas de Bancos de Dados, Editora Campus, 2004

HEUSER, C. A. Projeto de Banco de Dados, 5a. ed. Sagra Luzzato, 2004.

SILBERSCHATZ, A.; KORF, H. F.; SUDARSHAN, S. Sistemas de Banco de Dados, 6a.

Ed. Elsevier, 2012.

MYSQL, 2019