Isabelle Neves Porto

Apostila de Banco de Dados

Brasil
2021

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SGBD	Sistema Gerenciador de Banco de Dados
DBA	Database Administrator
SQL	Standard Query Language

Sumário

Introdução

Um banco de dados é uma coleção de dados. Um dado pode ser um fato sobre qualquer objeto em consideração (nome, idade e altura são dados relacionados a você), sendo um fato que deve ser armazenado/persistido e que tem um significado implícito.

Nós vamos estudar dois tipos de banco de dados: os relacionais e os não relacionais.

Os bancos relacionais são compostos por tabelas que relacionam os dados entre eles, sendo um exemplo de banco de dados relacional o PostgresSQL, o MySQL e o SQL Server.

Os bancos não relacionais utilizam uma estrutura de dados diferente do banco relacional, não dispondo de tabelas e sim coleções de dados no lugar. Nós veremos o que isso significa mais para frente. Exemplos de bancos não relacionais são o MongoDB, DynamoDB e Cassandra.

Nós chamamos de DBA (*Database Administrator*, administrador de banco de dados) o profissional que dentro de uma equipe de tecnologia é responsável por criar, manter e otimizar banco de dados.

Parte I

Bancos Relacionais

1 Introdução

Os dados armazenados nos bancos relacionais são organizados por tabelas. Essas tabelas possuem linhas e colunas, com cada célula representando o valor de um dado. Uma linha pode ser chamada de registro e contém uma instância exclusiva de dados (FERNANDES, 2020). Um exemplo de representação de uma tabela pode ser visto na figura 1.

Figura 1 – Uma tabela de banco de dados relacional

Existem softwares que criam isso no mundo computacional. Da mesma forma que existe o Excel, que salva dados em planilhas, existe os SGBD (Sistema Gerenciador de Banco de Dados), que manipulam banco de dados relacionais.

Um SGBD é um software que permite que seu usuário crie e mantenha um banco de dados. É um sistema que permite duas ações principais:

- **Definir um banco de dados:** criar as estruturas para armazenamento dos dados e especificar as restrições que devem ser impostas aos dados;
- **Manipular os dados dos banco de dados:** consultar, inserir, alterar e excluir dados do banco de dados.

Nessa apostila nós iremos nos basear no funcionamento do SGBD PostgreSQL. Para baixá-lo, visite esse site.

2 SQL

SQL significa Standard Query Language, e é uma linguagem padrão para trabalhar com bancos de dados relacionais. Cada SGBD tem sua implementação do SQL, mas de forma geral, os comandos são muito parecidos entre todos os bancos relacionais.

Nós chamamos de *query* uma consulta feita ao banco de dados, através da linguagem SQL. Exemplos de *queries* são listagem de dados, atualização de dados e deleção de dados.

2.1 *Queries* de criação de tabelas

Para passar a salvar os dados em uma tabela, nós precisamos primeiro definir essa tabela para o SGBD. Isso é feito através do comando CREATE TABLE, como exemplificado abaixo:

```
CREATE TABLE pessoa (
    id INTEGER PRIMARY KEY,
    nome VARCHAR(50)
);

CREATE TABLE disciplina (
    nome VARCHAR(50),
    carga_horaria INTEGER
);
```

A estrutura do comando é a seguinte: após o "CREATE TABLE", você coloca o nome que você quer dar à sua tabela. Dentro dos parenteses você vai declarar as colunas dessa sua tabela, escrevendo o nome da coluna seguido pelo tipo de dado que ela vai armazenar. Os tipos de dados disponíveis mais comuns são:

Tipo de dado	**Significado**
INTEGER	Armazena números inteiros
NUMERIC	Armazena números fracionários
VARCHAR(N)	Armazena caracteres com o tamanho máximo definido (N)
TEXT	Armazena caracteres sem limite de tamanho
DATE	Armazena uma data
TIMESTAMP	Armazena uma data com horário
BOOLEAN	Armazena um valor booleano (verdadeiro ou falso)

Tabela 1 – Tabela com lista dos tipos de dados mais comuns no PostgreSQL

Não pode existir duas tabelas com o mesmo nome, portanto caso você tente executar o mesmo comando de criação de tabela duas vezes, ele vai falhar na segunda vez. Também não é possível ter duas colunas na mesma tabela com o mesmo nome.

Na tabela pessoa, um dos campos está marcado como PRIMARY KEY. Isso significa que esse campo vai ser a chave primária dessa tabela. A chave primária de uma tabela é a coluna que possui o valor que vai identificar a instância de dados dentro da tabela, possuindo um valor único para cada linha. Funciona da mesma forma que o CPF para as pessoas no Brasil: é um valor único que identifica quem é essa pessoa. Não é possível inserir no banco de dados duas linhas em uma tabela com o mesmo valor para a chave primária: fazer isso seria como repetir o CPF de alguém para outra pessoa.

2.2 *Queries* de inserção de dados

Para inserir dados em uma tabela, é usado o comando "INSERT INTO". Exemplo usando as tabelas citadas anteriormente:

```
INSERT INTO pessoa VALUES (1, 'Isabelle');
INSERT INTO disciplina VALUES ('Back-end', 150);
```

Dentro de "VALUES" é colocado o valor que corresponde a cada coluna. Nesse caso, é necessário passar os valores na ordem que as colunas foram criadas. Caso você não queira usar a mesma ordem ou não queira preencher alguma coluna específica, pode executar o comando especificando as colunas dessa forma:

```
INSERT INTO pessoa(id) VALUES (2);
INSERT INTO disciplina(carga_horaria, nome) VALUES (100, 'Banco de dados');
```

2.3 *Queries* de seleção de dados

Para selecionar os dados de uma tabela, nós usamos o comando SELECT, seguindo a sintaxe abaixo:

```
SELECT * FROM tabela;
```

Esse comando irá trazer todos os dados da tabela chamada "tabela". O asterisco (*) apresentado na *query* representa a seleção de todas as colunas da tabela. Para selecionar colunas específicas, nós podemos fazer o seguinte:

```
SELECT id, nome FROM pessoa;
SELECT carga_horaria FROM disciplina;
```

Dessa forma, nós ainda estamos trazendo todos as linhas da tabela, porém agora estamos trazendo colunas específicas. Para especificar linhas da tabelas, nós podemos estipular condições. Essas condições são implementadas no SQL através da cláusula WHERE. Um exemplo seria:

```
SELECT * FROM pessoa WHERE nome = 'Isabelle';
SELECT id FROM disciplina WHERE carga_horaria > 100;
```

No primeiro exemplo nós estamos listando todos os dados das pessoas com o nome igual a Isabelle. No segundo exemplo nós estamos listando o id das disciplinas com carga horária maior que 100. Os operadores que nós podemos usar para criar condições são:

- $=$ restringe para linhas com o valor igual ao especificado;

  ```
  SELECT * FROM pessoa WHERE nome = 'Isabelle';
  ```

- $>$ **e** $<$ usado para colunas numéricas, restringe a valores maiores/menores que o especificado;

  ```
  SELECT * FROM disciplina WHERE carga_horaria > 100;
  ```

- $>=$ **e** $<=$ usado para colunas numéricas, restringe a valores maiores ou iguais ao especificado;

  ```
  SELECT * FROM disciplina WHERE carga_horaria <= 100;
  ```

- **IN** usado para restringir a um conjunto de valores;

  ```
  SELECT * FROM disciplina WHERE nome IN ('Banco de dados', 'Back-end');
  ```

Além de ter condições simples para a consulta, nós podemos também mesclar essas condições, através dos operadores AND e OR.

```
SELECT * FROM pessoa WHERE idade > 18 AND nome = 'Isabelle';
SELECT * FROM disciplina WHERE nome = 'Back-end' OR carga_horaria = 100;
```

Na primeira consulta nós estamos buscando por todas as pessoas com nome igual a "Isabelle" e com idade acima de 18. Na segunda consulta nós estamos buscando por todas as disciplinas cujo nome é igual a "Back-end" ou a carga horária é maior que 100, retornando todas as que baterem com pelo menos uma das condições especificadas.

Além dos operadores já apresentados, nós temos também as chamadas "funções de agregação". Funções de agregação são o que permite a execução de uma operação aritmética nos valores de uma coluna considerando todos os registros de uma tabela (REIS, 2013). As funções de agregação mais utilizadas são as seguintes:

- **COUNT:** conta quantos registros essa consulta retornaria. No exemplo, o retorno vai ser o número de pessoas com o nome "Isabelle":

```
SELECT COUNT(*) FROM pessoa WHERE nome = 'Isabelle';
```

- **MAX e MIN:** usado para colunas numéricas, retorna qual o valor máximo/mínimo que a coluna atingiu dentre todos os registros que correspondem com a condição especificada. No exemplo, irá ser retornado qual é o valor máximo de carga horária entre todas as disciplinas:

```
SELECT MAX(carga_horaria) FROM disciplina;
```

- **AVG:** usado para colunas numéricas, ele serve para calcular a média do valor de uma coluna dentre todas as linhas que correspondem com a condição especificada. No exemplo, será calculado a média de idade das pessoas com o nome "Isabelle":

```
SELECT AVG(carga_horaria) FROM disciplina;
```

- **SUM:** usado para colunas numéricas, retorna a soma dos valores de determinada coluna de todos os registros que correspondem com a condição especificada. O exemplo retorna a soma da carga horária das matérias de nome "Banco de dados" e "Back-end":

```
SELECT SUM(carga_horaria) FROM disciplina WHERE nome IN ('Banco de
dados', 'Back-end');
```

Referências

FERNANDES, H. M. *Banco de Dados Relacional (SQL) e Não Relacional (NoSQL) – O que são, para que servem e qual a diferença?* 2020. Disponível em: <https://marquesfernandes.com/tecnologia/banco-de-dados-relacional-sql-e-nao-relacional-nosql-o-que-sao-para-que-servem-e-qual-a-diferenca/>. Acesso em: 18.03.2021. Citado na página 7.

REIS, F. dos. *MySQL – Funções de Agregação (MAX, MIN, AVG, COUNT, SUM).* 2013. Disponível em: <http://www.bosontreinamentos.com.br/mysql/mysql-funcoes-de-agregacao-max-min-avg-count-sum-18/>. Acesso em: 18.03.2021. Citado na página 11.