Tabelas

Criação de tabelas

Para criar uma tabela no PostgreSQL, você precisa usar a linguagem DDL (Data Definition Language) e a sintaxe é a seguinte:

```
CREATE TABLE nome_da_tabela (
  nome_do_campo1 tipo_do_campo1,
  nome_do_campo2 tipo_do_campo2,
  ...,
  nome_do_campoN tipo_do_campoN
);
```

Por exemplo, para criar uma tabela chamada "clientes" com dois campos, "id" e "nome", o comando seria o seguinte:

```
CREATE TABLE clientes (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(50)
);
```

Este comando cria uma tabela "clientes" com um campo "id" que é um número serial (ou seja, um número sequencial gerado automaticamente pelo PostgreSQL) e um campo "nome" que é uma string de até 50 caracteres.

> Não se preocupe com o "PRIMARY KEY" ainda, isso será abordado nos próximos capítulos.

Edição de tabelas

Para editar uma tabela existente no PostgreSQL, você pode usar o comando ALTER TABLE. Por exemplo, se você quiser adicionar um novo campo "email" à tabela "clientes", o comando seria o seguinte:

```
ALTER TABLE clientes ADD COLUMN email VARCHAR(50);
```

Este comando adiciona um novo campo chamado "email" à tabela "clientes" com um comprimento máximo de 50 caracteres.

Exclusão de tabelas

Finalmente, para excluir uma tabela no PostgreSQL, você pode usar o comando DROP TABLE. Por exemplo, se você quiser excluir a tabela "clientes", o comando seria o seguinte:

```
DROP TABLE clientes;
```

Este comando exclui a tabela "clientes" e todos os dados armazenados nela.

É importante notar que a exclusão de uma tabela é uma operação permanente e irrevogável. Portanto, sempre se certifique de ter feito um backup dos dados importantes antes de executar um comando DROP TABLE.

Tipos de dados mais comuns

Nos exemplos acima, falamos dos tipos `int` e `varchar`, porém, existem alguns outros tipos de dados que são possíveis de serem utilizados, que nos auxiliam para situações específicas.

Tipos numéricos:

INTEGER: armazena números inteiros com sinal. O tamanho padrão é de 4 bytes, mas pode ser especificado um tamanho maior ou menor. É usado para armazenar dados numéricos que representam quantidades inteiras, como a quantidade de itens em um pedido.

BIGINT: armazena números inteiros com sinal de 8 bytes. É usado para armazenar dados numéricos que representam quantidades maiores que o INTEGER, como o número de habitantes de uma cidade ou país.

NUMERIC: armazena números decimais com precisão arbitrária. É usado para armazenar valores monetários ou outros dados numéricos que requerem alta precisão.

DOUBLE PRECISION: armazena números de ponto flutuante de dupla precisão. É usado para armazenar dados numéricos que requerem alta precisão e uma grande faixa de valores, como as coordenadas geográficas de um local.

Tipos de texto

VARCHAR: armazena strings de comprimento variável. É usado para armazenar dados de texto de comprimento variável, como nomes de pessoas ou endereços de e-mail.

CHAR: armazena strings de comprimento fixo. É usado para armazenar dados de texto de comprimento fixo, como códigos postais ou números de identificação.

TEXT: armazena strings de comprimento variável sem limite. É usado para armazenar grandes volumes de dados de texto, como descrições de produtos ou comentários em um site.

É comum a utilização do tipo `varchar` na maioria dos casos, pois geralmente campos de tamanhos variados são armazenados como texto. O `varchar` armazena apenas os caracteres salvos, e deixa o resto do espaço vazio e disponível para outro dado.

Por exemplo: uma coluna do tipo `varchar(50)` que recebe o texto: 'Ada tecnologia', utiliza apenas 14 "espaços" na memória, e deixa os outros 36 disponíveis para outros registros. Ele ocupa menos espaço em memória, porém, em operações de atualização é penalizado, pois o banco de dados precisa "procurar" outro espaço em memória para realocar o conteúdo caso seja maior.

Já uma coluna do tipo `char(50)` que recebe o mesmo texto: 'Ada tecnologia', utiliza todos os 50 "espaços" disponíveis, portanto, ocupa mais espaço em memória. Em contrapartida, as operações de atualização são extremamente rápidas, pois a coluna já tem todo o espaço disponível "reservado" para ela.

Tipos de data e hora

DATE: armazena datas (ano, mês, dia). É usado para armazenar informações de datas, como datas de nascimento ou datas de eventos.

TIME: armazena horas do dia. É usado para armazenar informações de hora, como horários de início ou fim de eventos.

TIMESTAMP: armazena datas e horas com precisão de milissegundos. É usado para armazenar informações de datas e horas precisas, como timestamps de criação ou modificação de registros.

Tipos booleanos

BOOLEAN: armazena valores verdadeiro ou falso. É usado para armazenar dados booleanos, como se um usuário está conectado ou desconectado.

Tipos de array

ARRAY: armazena uma lista de valores de um tipo de dado específico. É usado para armazenar coleções de dados, como uma lista de itens em um pedido ou as cores favoritas de um usuário.

Tipos especiais

JSON e JSONB: armazena dados em formato JSON (JavaScript Object Notation), usado para armazenar dados semiestruturados que podem ser facilmente lidos por outras

aplicações ou sistemas.

UUID: armazena identificadores únicos universais (UUIDs), que são frequentemente usados em sistemas distribuídos para identificar de forma única recursos em um cluster.

Você pode ver todos os tipos disponíveis para utilização no PostgreSQL em: https://www.postgresql.org/docs/current/datatype.html

Default values

Em PostgreSQL, um valor padrão (ou "default value") é um valor que é automaticamente atribuído a uma coluna se nenhum valor é especificado para ela durante a inserção de dados.

Por exemplo, vamos supor que você tem uma tabela "clientes" com as colunas "id", "nome", "email" e "data de cadastro". Você pode definir um valor padrão para a coluna "data de cadastro" para que, quando um novo registro for adicionado à tabela, a data atual seja automaticamente inserida para essa coluna. Isso pode ser útil, pois evita a necessidade de inserir manualmente a data a cada vez que um novo registro é adicionado.

Aqui está um exemplo de como definir um valor padrão em PostgreSQL:

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    data_de_cadastro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Neste exemplo, a coluna "data_de_cadastro" é definida com um valor padrão de "CURRENT_TIMESTAMP", que é uma função que retorna a data e hora atual do sistema. Isso significa que, se nenhum valor for especificado para essa coluna durante a inserção de dados, o PostgreSQL irá automaticamente inserir a data e hora atual.

Outro exemplo pode ser de uma tabela de pedidos, onde a coluna "status" pode ter um valor padrão "pendente" para todos os novos pedidos:

```
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    descricao VARCHAR(100),
    valor DECIMAL(10,2),
    status VARCHAR(20) DEFAULT 'pendente'
);
```

Neste exemplo, a coluna "status" é definida com um valor padrão "pendente", que será automaticamente inserido se nenhum valor for especificado durante a inserção de dados.

Além disso, é importante observar que um valor padrão pode ser definido para qualquer tipo de coluna, incluindo texto, números, datas, booleanos etc. É uma ferramenta útil para garantir a consistência dos dados em uma tabela e simplificar a inserção de dados.

Referências e materiais complementares

https://www.postgresql.org/docs/current/sql-createtable.html

https://www.postgresql.org/docs/current/sql-droptable.html

https://www.postgresql.org/docs/current/sql-altertable.html

https://www.postgresql.org/docs/current/datatype.html

https://www.postgresql.org/docs/current/ddl-default.html