Banco de Dados 2

Nome: Carlos Daniel de Moura Santos - Matricula: 31811BSI022

Passando parâmetros para uma função de trigger no PostgreSQL

Passo 1: Criando uma tabela auxiliar

```
CREATE TABLE trigger_params (
  id SERIAL PRIMARY KEY,
  param1 INTEGER,
  param2 VARCHAR(255)
);
```

Passo 2: Inserindo os valores dos parâmetros

```
INSERT INTO trigger_params (param1, param2)
VALUES (123, 'foo');
```

Passo 3: Definindo a função de trigger

```
CREATE OR REPLACE FUNCTION trigger_function()
RETURNS TRIGGER AS $$
DECLARE
  param1 INTEGER;
  param2 VARCHAR(255);
BEGIN
  SELECT param1, param2 INTO param1, param2
  FROM trigger_params
  WHERE id = (SELECT MAX(id) FROM trigger_params);

  -- Utilize os valores dos parâmetros na função de trigger

  RETURN NEW; -- ou RETURN NULL; dependendo do tipo de trigger
END;
$$ LANGUAGE plpgsql;
```

Passo 4: Criando o gatilho

```
CREATE TRIGGER my_trigger
AFTER INSERT ON my_table
FOR EACH ROW
EXECUTE FUNCTION trigger_function();
```

Funções de Data no PostgreSQL

- `CURRENT_DATE`: Retorna a data atual.
- `CURRENT_TIME`: Retorna o horário atual.
- `CURRENT_TIMESTAMP`: Retorna a data e hora atuais.
- `EXTRACT`: Extrai uma parte específica de uma data ou hora.
- `AGE`: Calcula a diferença entre dois timestamps, retornando um intervalo de tempo.
- `DATE_PART`: Extrai uma parte específica de uma data ou hora.
- `TO_CHAR`: Converte um timestamp em uma string formatada com base em um formato especificado.
- `TO_TIMESTAMP`: Converte uma string em um timestamp com base em um formato especificado.
- `DATE_ADD`: Adiciona um intervalo de tempo a um timestamp.
- `DATE_SUB`: Subtrai um intervalo de tempo de um timestamp.
- `DATE_PART('epoch', timestamp)`: Retorna o timestamp em segundos desde o epoch (1º de janeiro de 1970).

`CURRENT_DATE`: Retorna data atual

```
Exemplo: Retornar registros com a data atual
SELECT *
FROM tabela
WHERE data = CURRENT_DATE;
```

Nesse exemplo, estamos selecionando todos os registros da tabela onde a coluna `data` é igual à data atual, que é obtida usando a função `CURRENT_DATE`. Isso é útil quando você deseja filtrar registros com base na data atual.

Você pode adaptar esse exemplo de acordo com sua estrutura de tabela e critérios de filtro. Certifique-se de substituir `tabela` pelo nome correto da sua tabela e `data` pela coluna correspondente.

Espero que isso esclareça o uso da função `CURRENT_DATE` no PostgreSQL!

`CURRENT_TIME`: Retorna o horário atual.

A função `CURRENT_TIME` no PostgreSQL retorna o horário atual. Aqui está um exemplo de como utilizá-la em uma consulta no PostgreSQL:

```
Exemplo: Selecionar registros com horário posterior ao atual
SELECT *
FROM tabela
WHERE horario > CURRENT_TIME;
```

Nesse exemplo, estamos selecionando todos os registros da tabela onde a coluna `horario` é posterior ao horário atual, obtido pela função `CURRENT_TIME`. Isso pode ser útil quando você precisa filtrar registros com base no horário atual.

Certifique-se de substituir `tabela` pelo nome correto da sua tabela e `horario` pela coluna que contém os horários relevantes.

`CURRENT_TIMESTAMP`: Retorna a data e hora atuais.

A função `CURRENT_TIMESTAMP` no PostgreSQL retorna a data e hora atuais. Aqui está um exemplo de como utilizá-la em uma consulta:

```
Exemplo: Selecionar registros com timestamp posterior ao atual
SELECT *
FROM tabela
WHERE timestamp > CURRENT_TIMESTAMP;
```

Nesse exemplo, estamos selecionando todos os registros da tabela onde a coluna `timestamp` é posterior ao timestamp atual, obtido pela função `CURRENT_TIMESTAMP`. Isso é útil quando você precisa filtrar registros com base no timestamp atual.

Certifique-se de substituir `tabela` pelo nome correto da sua tabela e `timestamp` pela coluna que contém os valores de timestamp relevantes.

`EXTRACT`: Extrai uma parte específica de uma data ou hora.

A função `EXTRACT` no PostgreSQL é usada para extrair uma parte específica de uma data ou hora. Aqui está um exemplo de como utilizá-la em uma consulta:

```
Exemplo: Extrair o ano de uma data
SELECT EXTRACT(YEAR FROM data) AS ano
FROM tabela;
```

Nesse exemplo, estamos selecionando a parte do ano da coluna `data` da tabela e renomeando-a para `ano`. Isso extrai apenas o ano da data fornecida. Você pode substituir `tabela` pelo nome correto da sua tabela e `data` pela coluna que contém os valores de data.

A função `EXTRACT` pode ser usada para extrair outras partes, como mês, dia, hora, minuto, segundo, entre outros. Basta substituir `YEAR` pelo qualificador apropriado na função `EXTRACT`.

`AGE`: Calcula a diferença entre dois timestamps, retornando um intervalo de tempo.

A função `AGE` no PostgreSQL é usada para calcular a diferença entre dois timestamps, retornando um intervalo de tempo. Aqui está um exemplo de como utilizá-la em uma consulta:

```
Exemplo: Calcular a idade com base na data de nascimento
SELECT AGE(data_nascimento) AS idade
FROM pessoas;
```

Nesse exemplo, estamos calculando a diferença entre a data de nascimento, armazenada na coluna `data_nascimento`, e a data atual. A função `AGE` retorna um intervalo de tempo representando a diferença entre os dois timestamps. O resultado é renomeado como `idade` na consulta.

A função `AGE` também pode ser usada para calcular a diferença entre dois timestamps específicos, fornecidos como argumentos. Por exemplo:

```
Exemplo: Calcular a diferença entre duas datas
SELECT AGE(timestamp1, timestamp2) AS diferenca
FROM tabela;
```

Nesse caso, substitua `timestamp1` e `timestamp2` pelos valores de timestamp que você deseja calcular a diferença.

Espero que isso ajude a entender como usar a função `AGE` no PostgreSQL para calcular a diferença entre timestamps e obter um intervalo de tempo!

`DATE_PART`: Extrai uma parte específica de uma data ou hora.

A função `DATE_PART` no PostgreSQL é usada para extrair uma parte específica de uma data ou hora. Ela retorna um valor numérico correspondente à parte especificada. Aqui está um exemplo de como utilizá-la em uma consulta:

```
Exemplo: Extrair o ano da data
SELECT DATE_PART('year', data_coluna) AS ano
FROM tabela;
```

Nesse exemplo, estamos usando a função `DATE_PART` para extrair o ano da coluna `data_coluna` em uma tabela. A parte específica que queremos extrair é o ano, então passamos o argumento `'year'` para a função. O resultado é retornado como a coluna `ano` na consulta.

A função `DATE_PART` também pode ser usada para extrair outras partes de uma data ou hora, como mês, dia, hora, minuto, segundo, entre outros. Basta substituir o argumento `'year'` pelo valor correspondente à parte que deseja extrair.

```
Exemplo: Extrair o mês da data
SELECT DATE_PART('month', data_coluna) AS mes
FROM tabela;
sqlCopy code-- Exemplo: Extrair o dia da data
SELECT DATE_PART('day', data_coluna) AS dia
FROM tabela;
```

Certifique-se de substituir `data_coluna` pelo nome da coluna que contém a data ou hora que você deseja extrair a parte específica.

`TO_CHAR`: Converte um timestamp em uma string formatada com base em um formato especificado.

A função `TO_CHAR` no PostgreSQL é usada para converter um timestamp em uma string formatada com base em um formato específico. Essa função é útil quando você deseja exibir um timestamp em um formato personalizado. Aqui está um exemplo de como usá-la:

```
Exemplo: Converter um timestamp para o formato 'DD/MM/YYYY HH:MI:SS'
SELECT TO_CHAR(timestamp_coluna, 'DD/MM/YYYY HH:MI:SS') AS timestamp_formatado
FROM tabela;
```

Nesse exemplo, estamos usando a função `TO_CHAR` para converter a coluna `timestamp_coluna` em um formato específico. Passamos dois argumentos para a função: o primeiro é o valor do timestamp que queremos formatar, e o segundo é a string que define o formato desejado. No exemplo acima, estamos formatando o timestamp no formato 'DD/MM/YYYY HH:MI:SS', onde 'DD' representa o dia, 'MM' o mês, 'YYYY' o ano, 'HH' as horas, 'MI' os minutos e 'SS' os segundos.

Você pode personalizar o formato conforme suas necessidades, alterando a sequência de caracteres no segundo argumento da função `TO_CHAR`. Consulte a documentação do PostgreSQL para obter mais informações sobre os padrões de formatação disponíveis.

Certifique-se de substituir `timestamp_coluna` pelo nome da coluna que contém o timestamp que você deseja formatar.

`TO_TIMESTAMP`: Converte uma string em um timestamp com base em um formato especificado.

A função `TO_TIMESTAMP` no PostgreSQL é usada para converter uma string em um timestamp com base em um formato específico. Essa função é útil quando você precisa inserir um valor de data/hora em uma coluna do tipo timestamp ou quando deseja converter uma string em um valor de timestamp para realizar cálculos ou comparações.

Aqui está um exemplo de como usar a função `TO_TIMESTAMP`:

```
Exemplo: Converter uma string em um timestamp
SELECT TO_TIMESTAMP('2023-05-01 10:30:00', 'YYYY-MM-DD HH:MI:SS') AS
timestamp_convertido;
```

Nesse exemplo, estamos convertendo a string `'2023-05-01 10:30:00'` em um valor de timestamp usando o formato `'YYYY-MM-DD HH:MI:SS'`. A string de entrada e o formato especificado devem corresponder exatamente para que a conversão seja bem-sucedida.

Certifique-se de ajustar a string de entrada e o formato especificado de acordo com suas necessidades. É importante seguir o padrão especificado corretamente para garantir a correta conversão da string em um valor de timestamp.

`DATE_ADD`: Adiciona um intervalo de tempo a um timestamp.

No PostgreSQL, a função `DATE_ADD` não existe nativamente. No entanto, você pode adicionar um intervalo de tempo a um timestamp usando a adição direta do intervalo à coluna de timestamp desejada.

Aqui está um exemplo de como adicionar um intervalo de tempo a um timestamp no PostgreSQL:

```
Exemplo: Adicionar 1 dia a um timestamp
SELECT CURRENT_TIMESTAMP + INTERVAL '1 day' AS timestamp_adicionado;
```

Nesse exemplo, estamos adicionando 1 dia ao timestamp atual usando a sintaxe `CURRENT_TIMESTAMP + INTERVAL '1 day'`. Você pode substituir o valor `'1 day'` por qualquer outro intervalo de tempo desejado, como `'2 hours'`, `'3 months'`, etc.

Certifique-se de ajustar o intervalo de tempo de acordo com suas necessidades. Você pode adicionar ou subtrair qualquer quantidade de dias, horas, minutos, meses, etc., ao timestamp usando a adição direta do intervalo.

`DATE_SUB`: Subtrai um intervalo de tempo de um timestamp.

No PostgreSQL, a função `DATE_SUB` não existe nativamente. No entanto, você pode subtrair um intervalo de tempo de um timestamp usando a subtração direta do intervalo da coluna de timestamp desejada.

Aqui está um exemplo de como subtrair um intervalo de tempo de um timestamp no PostgreSQL:

```
Exemplo: Subtrair 1 semana de um timestamp
SELECT CURRENT_TIMESTAMP - INTERVAL '1 week' AS timestamp_subtraido;
```

Nesse exemplo, estamos subtraindo 1 semana do timestamp atual usando a sintaxe `CURRENT_TIMESTAMP - INTERVAL '1 week'`. Você pode substituir o valor `'1 week'` por qualquer outro intervalo de tempo desejado, como `'2 hours'`, `'3 months'`, etc.

`DATE_PART('epoch', timestamp)`: Retorna o timestamp em segundos desde o epoch (1º de janeiro de 1970).

A função `DATE_PART('epoch', timestamp)` no PostgreSQL retorna o valor do timestamp em segundos desde o epoch, que é definido como 1º de janeiro de 1970. Isso é útil quando você deseja obter o valor de um timestamp em formato de segundos para realizar cálculos ou comparações.

Aqui está um exemplo de como usar a função `DATE_PART` para obter o valor de um timestamp em segundos desde o epoch:

```
Exemplo: Obter o valor do timestamp atual em segundos desde o epoch
SELECT DATE_PART('epoch', CURRENT_TIMESTAMP) AS segundos_desde_epoch;
```

Nesse exemplo, usamos `CURRENT_TIMESTAMP` para obter o timestamp atual e aplicamos a função `DATE_PART` para extrair o valor em segundos desde o epoch. O resultado é retornado como uma coluna chamada `segundos_desde_epoch`.

Você também pode usar essa função com outros timestamps que você possui em suas tabelas, substituindo `CURRENT_TIMESTAMP` pelo nome da coluna de timestamp desejada.