TRIM

Thema: TRIM – Zeichenketten bereinigen und Leerzeichen entfernen

Einführung

Die SQL-Funktion `TRIM` wird verwendet, um unerwünschte Leerzeichen oder andere spezifische Zeichen von der **linken**, **rechten** oder **beiden Seiten** einer Zeichenkette zu entfernen. Dies ist besonders nützlich, um Daten zu bereinigen, beispielsweise beim Import oder bei der Verarbeitung von Benutzereingaben.

In diesem Thema lernst du:

1. Die Grundlagen und Syntax von `TRIM`.
2. Die verschiedenen Varianten (z. B. Entfernung nur von Leerzeichen oder benutzerdefinierten Zeichen).
3. Praktische Anwendungsbeispiele mit PostgreSQL.

Syntax von TRIM

Die allgemeine Syntax für `TRIM` lautet:

SQL

```sql
TRIM([LEADING | TRAILING | BOTH] [FROM] string)

```

Erklärung:

- `LEADING`: Entfernt Zeichen nur **von der linken Seite** der Zeichenkette.
- `TRAILING`: Entfernt Zeichen nur **von der rechten Seite** der Zeichenkette.
- `BOTH` (Standard): Entfernt Zeichen **von beiden Seiten** der Zeichenkette.
- `string`: Die Zeichenkette, die bereinigt werden soll.
- `FROM` (optional): Kann verwendet werden, um die Lesbarkeit zu verbessern, ist aber nicht erforderlich.

Standardverhalten:

Wenn keine spezifische Anweisung (`LEADING`, `TRAILING`, `BOTH`) angegeben ist, entfernt `TRIM` automatisch Zeichen **von beiden Seiten**.

Varianten von TRIM

1. **Standard-TRIM: Entfernt Leerzeichen**

```sql
SELECT TRIM('  Hello World  ') AS result;
```

Ergebnis: `'Hello World'`

2. **Entfernen von spezifischen Zeichen** Du kannst mit `TRIM` auch **andere Zeichen** als Leerzeichen entfernen:

```sql
SELECT TRIM('x' FROM 'xxHello Worldxx') AS result;
```

Ergebnis: `'Hello World'`

3. **Nur von der linken Seite entfernen (`LEADING`)**

```sql
SELECT TRIM(LEADING '0' FROM '00012345') AS result;
```

Ergebnis: `'12345'`

4. **Nur von der rechten Seite entfernen (`TRAILING`)**

```sql
SELECT TRIM(TRAILING '.' FROM 'file_name....') AS result;
```

Ergebnis: `'file_name'`

5. **Leerzeichen explizit entfernen**

```sql
SELECT TRIM(BOTH ' ' FROM '   PostgreSQL   ') AS result;
```

Ergebnis: `'PostgreSQL'`

Praktische Anwendungsbeispiele

Beispiel 1: Bereinigung von Benutzereingaben

Angenommen, du hast eine Tabelle `users` mit einer Spalte `username`, und Benutzer haben unabsichtlich Leerzeichen in ihren Eingaben hinterlassen:

SQL

```sql
SELECT TRIM(username) AS cleaned_username
FROM users;
```

Vorher: `' alice '`

Nachher: `'alice'`

Beispiel 2: Entfernen von Präfixen

Ein Kunde hat Produkt-IDs mit einem unerwünschten Präfix, und du möchtest dieses Präfix entfernen:

SQL

```sql
SELECT TRIM(LEADING 'ID-' FROM 'ID-12345') AS
product_id;
```

Ergebnis: `'12345'`

Beispiel 3: Entfernen von spezifischen Zeichen in Kombination mit anderen Funktionen

Entferne Sonderzeichen und normalisiere die Eingaben auf Kleinbuchstaben:

SQL

```sql
SELECT LOWER(TRIM('_' FROM '_Data_Sample_')) AS
normalized_data;
```

Ergebnis: `'data_sample'`

Beispiel 4: Datenabgleich bei Leerzeichen

Vergleiche zwei Strings, die möglicherweise führende oder nachgestellte Leerzeichen enthalten:

SQL

```sql
SELECT
    CASE
        WHEN TRIM('  Hello ') = 'Hello' THEN 'Match'
        ELSE 'No Match'
    END AS comparison_result;

```

Ergebnis: `'Match'`

Übungen

1. Entferne alle führenden Nullen aus der Zeichenkette `'00009876'`.
2. Bereinige die Zeichenkette `'*Sample Data*'`, indem du die Sternchen (`*`) entfernst.
3. Bereinige und vergleiche zwei Zeichenketten, bei denen eine von beiden Leerzeichen enthält (`' Text '` und `'Text'`).

Best Practices

1. **Automatische Bereinigung von Eingabedaten:**

Entferne unerwünschte Leerzeichen direkt nach dem Datenimport oder vor dem Speichern in der Datenbank.

2. **Kombination mit anderen Funktionen:**

Verwende `TRIM` zusammen mit Funktionen wie `LOWER` oder `UPPER`, um Daten einheitlich zu formatieren.

3. **Speicherplatzoptimierung:**

Bei großen Datenmengen kann das Bereinigen von Leerzeichen die Speichereffizienz verbessern.

Mit `TRIM` kannst du Zeichenketten effizient bereinigen und in PostgreSQL saubere, konsistente Daten verwalten.