

## Konstanten anderer Typen

Konstanten anderer Typen Konstanten eines beliebigen Typs können folgendermassen geschrieben werden: `typ 'zeichen'` `'zeichen' :: typ` `CAST ('zeichen' AS typ)` Tatsächlich sind dies alles Zeichenkettenkonstanten mit ausdrücklicher Typumwandlung. Die Typumwandlung kann weggelassen werden, wenn es keine Zweideutigkeit bezüglich des Typs gibt, den die Konstante haben muss (zum Beispiel wenn sie als Argument einer nicht überladenen Funktion verwendet wird); in diesen Fall wird sie automatisch umgewandelt. Die Formen `::` und `CAST ()` können auch verwendet werden, um den Typ eines beliebigen Ausdrucks anzugeben, wohingegen `typ 'zeichen'` nur mit Konstanten verwendet werden kann. Eine weitere Einschränkung bezüglich `typ 'zeichen'` ist, dass es nicht für Arraytypen funktioniert. Verwenden Sie `::` oder `cast()`, um den Typ einer Arraykonstanten festzulegen. Operatoren Ein Operatorname ist eine Folge von bis zu 63 Zeichen aus der folgenden Liste. `+ - * / , . = ~ ! @ # % ^ & | ` ?` mit folgenden Einschränkungen: `>`, `--` und `/ *` können nirgendwo in einem Operatornamen auftauchen, weil sie als der Anfang eines Kommentars verstanden werden.

Ein Operatorname, der aus mehreren Zeichen besteht, kann nicht auf `+` oder `-` enden, es sei denn, der Name enthält mindestens eins der folgenden Zeichen. `~ ! @ # % ^ & | ` ?` So ist zum Beispiel `@-` ein zulässiger Operatorname, nicht aber `*-`. Diese Einschränkung erlaubt es PostgreSQL, Befehle, die dem SQL-Standard folgen, zu verarbeiten, ohne Leerzeichen zwischen den Tokens zu benötigen. Alle Operatornamen können als binärer Infix-Operator, unärer Präfix-Operator oder unärer Postfix-Operator definiert sein. Ausserdem können die Schlüsselwörter `AND`, `OR` und `NOT` als Operatoren gezählt werden.

Schemaqualifizierte Operatornamen müssen folgendermassen geschrieben werden:

`OPERATOR (schema.operator)`

Zum Beispiel: `SELECT 3 OPERATOR(pg_catalog.+) 4;` Diese Konstruktion ist allerdings nur für Ausnahmefälle gedacht; in der Praxis verwendet man normalerweise den Schemasuchpfad für Operatoren. Operatornamen, die nicht dem SQL-Standard entsprechen, müssen meist durch Leerzeichen voneinander getrennt werden, um Unklarheiten zu vermeiden. Wenn Sie zum Beispiel einen linken Präfixoperator namens `@` definiert haben, dann können Sie nicht `X*@Y` schreiben, sondern Sie müssen `X* @Y` (oder vielleicht besser `X*(@Y)`) schreiben, um sicherzustellen, dass PostgreSQL den Ausdruck als zwei Operatornamen liest und nicht als nur einen. Bei Operatoren, die dem SQL-Standard entsprechen, ist dies normalerweise wegen der zweiten obigen Regel nicht notwendig.

Die hier aufgeführten Syntaxregeln beschreiben nur die theoretisch möglichen Operatornamen.

Die folgende Tabelle zeigt den Vorrang (in absteigender Reihenfolge) und die Anhänglichkeit der Operatoren sowie anderer Syntax-elemente in PostgreSQL. Operator/Element Anhänglichkeit Beschreibung links Trennung von Tabellen-/Spaltennamen:: links Typumwandlung PostgreSQL-Stil[] links Auswahl Arrayelement-rechts unäres Minus (Minus als Vorzeichen)^ links Potenzierung\* / % links Multiplikation, Division, Modulus+ - links Addition, Subtraktion IS TRUE, IS FALSE, IS UNKNOWN, IS NULL IS NULL Test für Null NOT NULL Test für nicht NULL (alle anderen) links alle anderen eingebauten und benutzerdefinierten Operatoren sowie OPERATOR(. . .)-

Konstrukte IN Mitgliedschaft BETWEEN Vergleich OVERLAPS Zeitintervall überlappt LIKE ILIKE

SIMILAR Mustervergleich von Zeichenketten <, > kleiner als, grosser als = rechts Gleichheit,

Wertzuweisung NOT rechts logische Negierung AND links logische Konjunktion OR links logische Disjunktion Um den Vorrang der Operatoren in Ausdrücken zu ändern, müssen Klammern verwendet werden.