Combining and Filtering Data with PostgreSQL

WORKING WITH STRING FUNCTIONS

Jason P. Browning
EXECUTIVE DIRECTOR, DIXIE STATE UNIVERSITY
@jason_from_ky

Overview

Data comes in many forms

Stringing it together

Manipulating strings

Using string functions to clean and analyze data

Data Type

What type of value can be stored in a table column

SQL Data Types

String

Numeric

Date and Time

Other

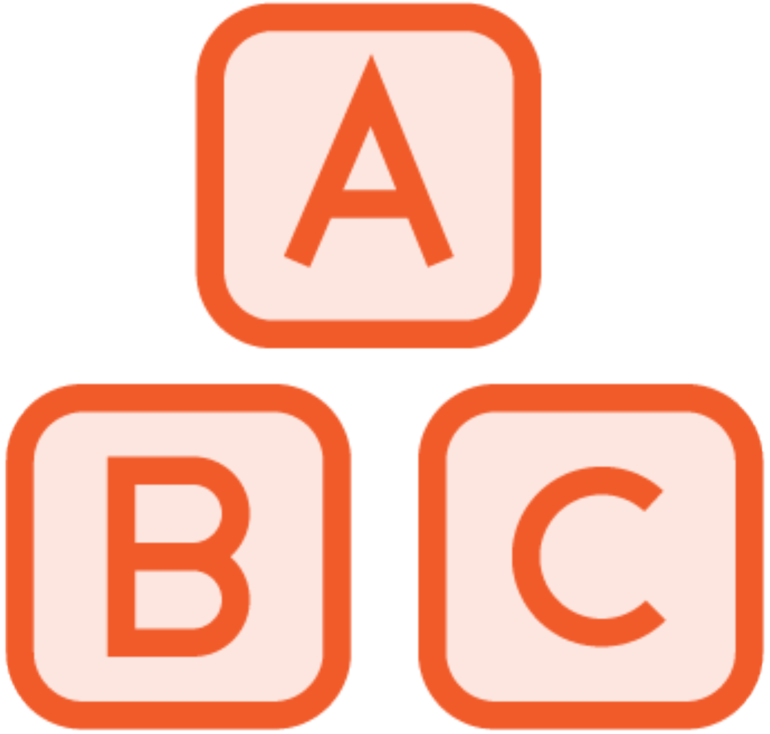
Data Types

Data types are not immutable

Can be changed or reassigned

Can be converted when writing query

String Data



Alphanumeric data

- Letters
- Numbers
- Special characters

Cannot perform calculations without conversion

String Data

CHAR

VARCHAR

TEXT

Concatenation

Joining multiple string fields together

Can specify additional text

Creates a single string text output

Concatenation Methods

Concatenation Operator	`field1 || ‘, ‘ || field2`
Concatenation Function	`CONCAT(field1, ‘, ‘, field2)`
With Separator Function	`CONCAT_WS(‘, ‘, field1, field2)`

Population Table

city	state	population
Boise	Idaho	226,570
Cincinnati	Ohio	301,301
Cleveland	Ohio	385,525
Louisville	Colorado	21,128
Louisville	Kentucky	616,261

```
SELECT city || ‘, ‘ || state
         AS location,
       population
  FROM population
 WHERE city = ‘Louisville’;
```

- Use the string concatenation operator || to combine multiple fields
- This method complies with ANSI SQL

Location	Population
Louisville, Colorado	21,128
Louisville, Kentucky	616,261

```
SELECT CONCAT(city, ‘,’, state)
        AS location,
      population
 FROM population
WHERE city = ‘Louisville’;
```

- Use the string concatenation function CONCAT to combine multiple fields

Location	Population
Louisville, Colorado	21,128
Louisville, Kentucky	616,261

```
SELECT CONCAT_WS(‘,’, city, state)
          AS location,
       population
  FROM population
 WHERE city = ‘Louisville’;
```

- Use the string concatenation function CONCAT_WS to combine multiple fields

Location	Population
Louisville, Colorado	21,128
Louisville, Kentucky	616,261

String Manipulation Functions

Field trimming and whitespace

Isolating part of a string

Changing case or characters

Field Trimming and Whitespace

```
> SELECT TRIM(‘ radar ‘);

      ‘radar’


> SELECT TRIM(‘r’ FROM ‘radar’);

      ‘ada’


> SELECT TRIM(LEADING ‘r’ FROM ‘radar’);

      ‘adar’


> SELECT TRIM(TRAILING ‘a’ FROM ‘radar’);

      ‘rada’
```

- TRIM removes whitespace or specified characters from string
- Function defaults remove whitespace before/after string
- Optional leading or trailing as first argument
- Can specify specific characters to remove

Isolating Parts of a String

LEFT

LEFT(string, n)

SPLIT_PART

SPLIT_PART(string, delimiter, field number)

RIGHT

RIGHT(string, n)

SUBSTRING

SUBSTRING(string, start position, length)

```
> SELECT LEFT('Pluralsight',6);

      'Plural'


> SELECT RIGHT('Pluralsight', 5);

      'sight'
```

LEFT and RIGHT

Specify which side of the field to start counting

First argument is the field

Second argument is the number of characters to return

```
> SELECT SPLIT_PART('USA/DC/202','/',2);

      'DC'


> SELECT SUBSTRING('USA/DC/202',5,2);

      'DC'

> SELECT SUBSTRING('USA/DC/202' FROM 5
    FOR 2);

      'DC'


> SELECT SUBSTRING('USA/DC/202',5);

      'DC/202'
```

SPLIT_PART

- Split field into components
- Based on delimiter

SUBSTRING

- Return substring from field
- Specify starting position and length
- Alternate syntax
  "FROM starting position FOR length"
- If length is not specified, will return from starting position to end of string

Changing Case

john doe

LOWER

SELECT LOWER(name)

JOHN DOE

UPPER

SELECT UPPER(name)

John Doe

INITCAP

SELECT INITCAP(name)

Changing Characters

REPLACE

REVERSE

```
SELECT REPLACE(street, 'Ave.', 'Avenue')
       AS street,
       city
  FROM address;
```

street	city
Hill Avenue	Boise
Main Street	Cincinnati
Sheridan Avenue	Cleveland
Cherokee Square	Louisville
Newburg Road	Louisville

Address table

street	city
Hill Ave.	Boise
Main Street	Cincinnati
Sheridan Ave.	Cleveland
Cherokee Square	Louisville
Newburg Road	Louisville

REPLACE
- Search for a substring
- Replace with a new substring

REVERSE

```
SELECT street,
      REVERSE(city) AS city
  FROM address;
```

street	city
Hill Ave.	esioB
Main Street	itannicniC
Sheridan Ave.	dnalevelC
Cherokee Square	ellivsiuoL
Newburg Road	ellivsiuoL

Address table

street	city
Hill Ave.	Boise
Main Street	Cincinnati
Sheridan Ave.	Cleveland
Cherokee Square	Louisville
Newburg Road	Louisville

REPLACE

REVERSE

- Arrange a string in reverse order

Demo

Applying string functions

Summary

Relational databases contain many data types

Strings are alphanumeric data

Use string functions to manipulate data to a more useful format for presentation and analysis