[Объяснение строковых функций Spark SQL](https://sparkbyexamples.com/spark/usage-of-spark-sql-string-functions/)

Spark SQL определяет встроенные стандартные строковые функции в DataFrame API. Эти строковые функции пригодятся, когда нам нужно выполнять операции со строками. В этой статье мы изучим использование некоторых функций на примере Scala. Вы можете получить доступ к стандартным функциям, используя следующий оператор импорта.

*// Импортировать функции Spark sql*

*import org.apache.spark.sql.functions.\_*

Когда это возможно, старайтесь использовать стандартные библиотечные функции, поскольку они немного более безопасны во время компиляции, обрабатывают null и работают лучше по сравнению с пользовательскими функциями. Если ваше приложение имеет решающее значение для производительности, постарайтесь любой ценой избегать использования пользовательских функций UDF, поскольку они не гарантируют производительность.

Связанный: Если вы ищете PySpark, обратитесь к разделу [Строковые функции PySpark SQL.](https://sparkbyexamples.com/pyspark/pyspark-string-functions-with-examples/)

Строковые функции Spark SQL:

Нажмите на каждую ссылку в таблице ниже, чтобы получить дополнительные объяснения и рабочие примеры строковой функции на примере Scala.

|  |  |
| --- | --- |
| СТРОЧНАЯ ФУНКЦИЯ НАЗВАНИЕ | ОПИСАНИЕ |
| ascii(e: Column): Column | Вычисляет числовое значение первого символа строкового столбца и возвращает результат в виде столбца типа int. |
| base64(e: Column): Column | Вычисляет кодировку BASE64 двоичного столбца и возвращает его как строковый столбец. Это противоположность unbase64. |
| concat\_ws(sep: String, exprs: Column\*): Column | Объединяет несколько столбцов входной строки в один столбец строки, используя заданный разделитель. |
| decode(value: Column, charset: String): Column | Вычисляет первый аргумент в строку из двоичного файла, используя предоставленный набор символов (один из «US-ASCII», «ISO-8859-1», «UTF-8», «UTF-16BE», «UTF-16LE», «UTF-16»). |
| encode(value: Column, charset: String): Column | Вычисляет первый аргумент в двоичном формате из строки, используя предоставленный набор символов (один из «US-ASCII», «ISO-8859-1», «UTF-8», «UTF-16BE», «UTF-16LE», «UTF-16»). |
| format\_number(x: Column, d: Int): Column | Форматирует числовой столбец x в формате «#,###,###.##», округляет его до d десятичных знаков в режиме округления HALF\_EVEN и возвращает результат в виде строкового столбца. |
| format\_string(format: String, arguments: Column\*): Column | Форматирует аргументы в стиле printf и возвращает результат в виде строкового столбца. |
| initcap(e: Column): Column | Возвращает новый строковый столбец, преобразуя первую букву каждого слова в верхний регистр. Слова разделяются пробелами. Например, «hello world» станет «Hello World». |
| instr(str: Column, substring: String): Column | Найдите позицию первого вхождения столбца substr в данной строке. Возвращает значение NULL, если любой из аргументов имеет значение NULL. |
| length(e: Column): Column | Вычисляет длину символов заданной строки или количество байтов двоичной строки. Длина строк символов включает конечные пробелы. Длина двоичных строк включает двоичные нули. |
| lower(e: Column): Column | Преобразует строковый столбец в нижний регистр. |
| levenshtein ( l : Column , r : Column ) : Column | Вычисляет расстояние Левенштейна для двух заданных строковых столбцов. |
| locate(substr: String, str: Column): Column | Найдите позицию первого вхождения substr. |
| locate(substr: String, str: Column, pos: Int): Column | Найдите позицию первого вхождения substr в строковом столбце после позиции pos. |
| lpad(str: Column, len: Int, pad: String): Column | Удерживая левую кнопку мыши, дополните столбец строки до длины len. Если строковый столбец длиннее, чем len, возвращаемое значение сокращается до len символов. |
| ltrim(e: Column): Column | Обрезать пробелы с левого конца для указанного строкового значения. |
| regexp\_extract(e: Column, exp: String, groupIdx: Int): Column | Извлеките определенную группу, соответствующую регулярному выражению Java, из указанного строкового столбца. Если регулярное выражение не соответствует или указанная группа не соответствует, возвращается пустая строка. |
| regexp\_replace(e: Column, pattern: String, replacement: String): Column | Заменить все подстроки указанного строкового значения, соответствующие регулярному выражению, на Rep. |
| regexp\_replace(e: Column, pattern: Column, replacement: Column): Column | Заменить все подстроки указанного строкового значения, соответствующие регулярному выражению, на Rep. |
| unbase64(e: Column): Column | Декодирует строковый столбец в кодировке BASE64 и возвращает его как двоичный столбец. Это противоположность base64. |
| rpad(str: Column, len: Int, pad: String): Column | Заполните столбец строки правой кнопкой мыши до длины len. Если строковый столбец длиннее, чем len, возвращаемое значение сокращается до len символов. |
| repeat(str: Column, n: Int): Column | Повторяет строковый столбец n раз и возвращает его как новый строковый столбец. |
| rtrim(e: Column): Column | Обрезать пробелы с правого конца для указанного строкового значения. |
| rtrim(e: Column, trimString: String): Column | Обрезать указанную строку символов с правого конца для указанного столбца строки. |
| soundex(e: Column): Column | Возвращает код soundex для указанного выражения |
| split(str: Column, regex: String): Column | Разбивает строку вокруг совпадений данного регулярного выражения. |
| split(str: Column, regex: String, limit: Int): Column | Разбивает строку вокруг совпадений данного регулярного выражения. |
| substring(str: Column, pos: Int, len: Int): Column | Подстрока начинается с pos и имеет длину len, если str имеет тип String, или возвращает фрагмент массива байтов, который начинается с pos в байтах и ​​имеет длину len, если str имеет двоичный тип. |
| substring\_index(str: Column, delim: String, count: Int): Column | Возвращает подстроку из строки str до подсчета вхождений разделителя. \* Если счетчик положителен, возвращается все, что находится слева от последнего разделителя (считая слева). Если счетчик отрицательный, возвращается каждый элемент справа от последнего разделителя (считая начиная с \* справа). substring\_index выполняет сопоставление с учетом регистра при поиске разделителя. |
| overlay(src: Column, replaceString: String, pos: Int, len: Int): Column | Наложите указанную часть `src` на `replaceString`, \* начиная с позиции байта `pos` в `inputString` и продолжая для `len` байтов. |
| overlay(src: Column, replaceString: String, pos: Int): Column | Наложите указанную часть `src` на `replaceString`, \* начиная с позиции байта `pos` в `inputString`. |
| translate(src: Column, matchingString: String, replaceString: String): Column | \* Символы в replaceString соответствуют символам в matchStrin. Переведите любой символ в src на символ в replaceString. г. \* Перевод произойдет, когда любой символ в строке соответствует символу \* в `matchingString`. |
| trim(e: Column): Column | Обрежьте пробелы с обоих концов для указанного строкового столбца. |
| trim(e: Column, trimString: String): Column | Обрезать указанный символ с обоих концов для указанного строкового столбца. |
| upper(e: Column): Column | Преобразует строковый столбец в верхний регистр. |

Заключение:

В этом посте я объединил полный список строковых функций Spark SQL с описанием и примерами некоторых часто используемых функций. Дополнительную информацию об этом можно найти в [следующем блоге.](https://databricks.com/blog/2015/09/16/apache-spark-1-5-dataframe-api-highlights.html)

Статьи по Теме

1. [Полный список функций массива Spark SQL](https://sparkbyexamples.com/spark/spark-sql-array-functions/)
2. [Встроенные стандартные функции Spark SQL](https://sparkbyexamples.com/spark/spark-sql-functions/)
3. [Функции Spark SQL Sort – полный список](https://sparkbyexamples.com/spark/spark-sql-sort-functions/)
4. [Spark SQL Like() с использованием примера подстановочных знаков](https://sparkbyexamples.com/spark/spark-sql-like-using-wildcard-example/)
5. [Функции даты и отметки времени Spark SQL](https://sparkbyexamples.com/spark/spark-sql-date-and-time-functions/)
6. [Spark SQL – выбор столбцов из DataFrame](https://sparkbyexamples.com/spark/spark-select-columns-from-dataframe/)
7. [Внутреннее соединение Spark SQL с примером](https://sparkbyexamples.com/scala/spark-sql-inner-join-with-example/)