[Полный список функций массива Spark SQL](https://sparkbyexamples.com/spark/spark-sql-array-functions/)

Spark SQL предоставляет встроенные стандартные функции массива, определенные в API DataFrame, они пригодятся, когда нам нужно выполнять операции над столбцом массива ([ArrayType](https://sparkbyexamples.com/spark/spark-array-arraytype-dataframe-column/)). Все они принимают входные данные как столбец массива и несколько других аргументов, основанных на функции.

Когда это возможно, старайтесь использовать стандартную библиотеку, поскольку она немного более безопасна во время компиляции, обрабатывает нулевые значения и работает лучше по сравнению с UDF. Если ваше приложение имеет решающее значение для производительности, старайтесь любой ценой избегать использования пользовательских UDF, поскольку они не гарантируют производительность.

Функции массива Spark SQL сгруппированы как функции сбора «collection\_funcs» в Spark SQL вместе с несколькими функциями сопоставления. Эти функции массива пригодятся, когда мы хотим выполнить некоторые операции и преобразования над столбцами массива.

Хотя я объяснял здесь использование Scala, аналогичные методы можно использовать для работы функции массива Spark SQL с PySpark, и если позволит время, я расскажу об этом в будущем. Если вы ищете PySpark, я все равно рекомендую прочитать эту статью, поскольку она даст вам представление о функциях и использовании массива Spark.

# Функции массива Spark SQL:

|  |  |
| --- | --- |
| СИНТАКСИС ФУНКЦИИ МАССИВА | ОПИСАНИЕ ФУНКЦИИ МАССИВА |
| array\_contains(column: Column, value: Any) | Проверьте, присутствует ли значение в столбце массива. Вернет значения ниже.  true – Возвращает, если значение присутствует в массиве.  false – когда присутствует значение eno.  null – когда массив равен нулю. |
| array\_distinct(e: Column) | Возвращайте отдельные значения из массива после удаления дубликатов. |
| array\_except(col1: Column, col2: Column) | Возвращает все элементы из массива col1, но не из массива col2. |
| array\_intersect(col1: Column, col2: Column) | Возвращает все элементы, присутствующие в массивах col1 и col2 |
| array\_join(column: Column, delimiter: String, nullReplacement: String)  array\_join(column: Column, delimiter: String) | Объединяет все элементы столбца массива с использованием предоставленного разделителя. Если присутствуют значения Null, они заменяются строкой nullReplacement. |
| array\_max(e: Column) | Возвращать максимальные значения в массиве |
| array\_min(e: Column) | Возвращать минимальные значения в массиве |
| array\_position(column: Column, value: Any) | Возвращает позицию/индекс первого вхождения «значения» в данном массиве. Возвращает позицию как длинный тип, и позиция не отсчитывается от нуля, а начинается с 1.Возвращает ноль, если значение не найдено.Возвращает значение NULL, если любой из аргументов имеет значение NULL. |
| array\_remove(column: Column, element: Any) | Возвращает массив после удаления всех предоставленных «значений» из данного массива. |
| array\_repeat(e: Column, count: Int) | Создает массив, содержащий первый аргумент, повторяемый количество раз, заданное вторым аргументом. |
| array\_repeat(left: Column, right: Column) | Создает массив, содержащий первый аргумент, повторяемый количество раз, заданное вторым аргументом. |
| array\_sort(e: Column) | Возвращает отсортированный массив данного входного массива. Все нулевые значения помещаются в конец массива. |
| array\_union(col1: Column, col2: Column) | Возвращает массив элементов, присутствующих в обоих массивах (все элементы из обоих массивов), без дубликатов. |
| arrays\_overlap(a1: Column, a2: Column) | true – если `a1` и `a2` имеют хотя бы один общий ненулевой элемент  false – если `a1` и `a2` имеют совершенно разные элементы.  null – если оба массива не пусты и любой из них содержит `null` |
| arrays\_zip(e: Column\*) | Возвращает объединенный массив структур, в котором N-я структура содержит все N-е входные значения. |
| concat(exprs: Column\*) | Объединяет все элементы из заданных столбцов |
| element\_at(column: Column, value: Any) | Возвращает элемент массива, расположенный в позиции ввода «значение». |
| exists(column: Column, f: Column => Column) | Проверяет, присутствует ли столбец в столбце массива. |
| explode(e: Column) | Создайте строку для каждого элемента в столбце массива. |
| explode\_outer ( e : Column ) | Создайте строку для каждого элемента в столбце массива. В отличие от разнесения, если массив имеет значение NULL или пуст, он возвращает значение NULL. |
| filter(column: Column, f: Column => Column)  filter(column: Column, f: (Column, Column) => Column) | Возвращает массив элементов, для которых содержится предикат в данном массиве. |
| flatten(e: Column) | Создает один массив из массива массивов столбцов. |
| forall(column: Column, f: Column => Column) | Возвращает, выполняется ли предикат для каждого элемента массива. |
| posexplode(e: Column) | Создает строку для каждого элемента массива и создает два столбца «pos» для хранения положения элемента массива и «col» для хранения фактического значения массива. |
| posexplode\_outer(e: Column) | Создает строку для каждого элемента массива и создает два столбца «pos» для хранения положения элемента массива и «col» для хранения фактического значения массива. В отличие от Posexplode, если массив имеет значение NULL или пуст, он возвращает значение NULL,NULL для столбцов pos и col. |
| reverse(e: Column) | Возвращает массив элементов в обратном порядке. |
| sequence(start: Column, stop: Column) | Сгенерируйте последовательность чисел от начала до конца. |
| sequence ( start : Column , stop : Column , step : Column ) | Сгенерируйте последовательность чисел от начала до конца, увеличивая заданное значение шага. |
| shuffle(e: Column) | Перемешать данный массив |
| size(e: Column) | Возвращает длину массива. |
| slice(x: Column, start: Int, length: Int) | Возвращает массив элементов из позиции «начало» и заданной длины. |
| sort\_array(e: Column) | Сортирует массив в порядке возрастания. Нулевые значения размещаются в начале. |
| sort\_array(e: Column, asc: Boolean) | Сортирует массив по возрастанию или убыванию на основе логического параметра. При присвоении нулевые значения помещаются в начале. А для десендинга это места в конце. |
| transform(column: Column, f: Column => Column)  transform(column: Column, f: (Column, Column) => Column) | Возвращает массив элементов после применения преобразования. |
| zip\_with(left: Column, right: Column, f: (Column, Column) => Column) | Объединяет два входных массива. |
| aggregate(  expr: Column,  zero: Column,  merge: (Column, Column) => Column,  finish: Column => Column) | Агрегация |

Статьи по Теме

1. Объяснение строковых функций Spark SQL
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/)