**Title:** Redis SUNION

***Excerpt:*** The SUNION command is used to get the union of one or more sets. It operates on linear time complexity. The SUNION command returns a list of distinct members that belong to the specified sets in the command.

**Permalink:** redis-sunion

**Category:** Redis

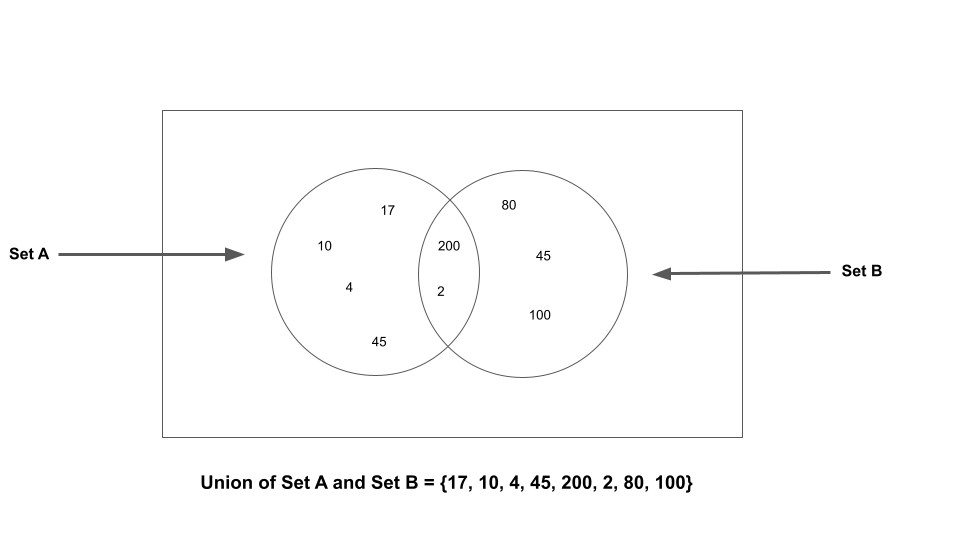
# 

# 

# 

# Union of Sets

Set can consist of a finite collection of alphabets, numerical values, or even real-world objects. In number systems, we have a set of operations to perform mathematical calculations. Similarly, set operations are needed when we need to establish a relationship between two or more sets. In this guide, we will be focusing only on set Union operations. Set union is defined as the distinct elements that are in set A, in set B, or both sets A and B where A and B are two given sets.



As shown in the above figure, all the distinct elements belong to set A and Set B, or both are called the union of set A and set B.

# Redis Sets and SUNION Command

Redis in-memory data store supports set data types to hold a collection of unique string members in an unordered manner. It supports major set operations such as union, intersection, and difference as well. In this discussion, we will be focusing on the union operation in Redis sets. The SUNION command is used to get the union of given sets. It has O(N) linear time complexity where N is the number of members in all the sets.

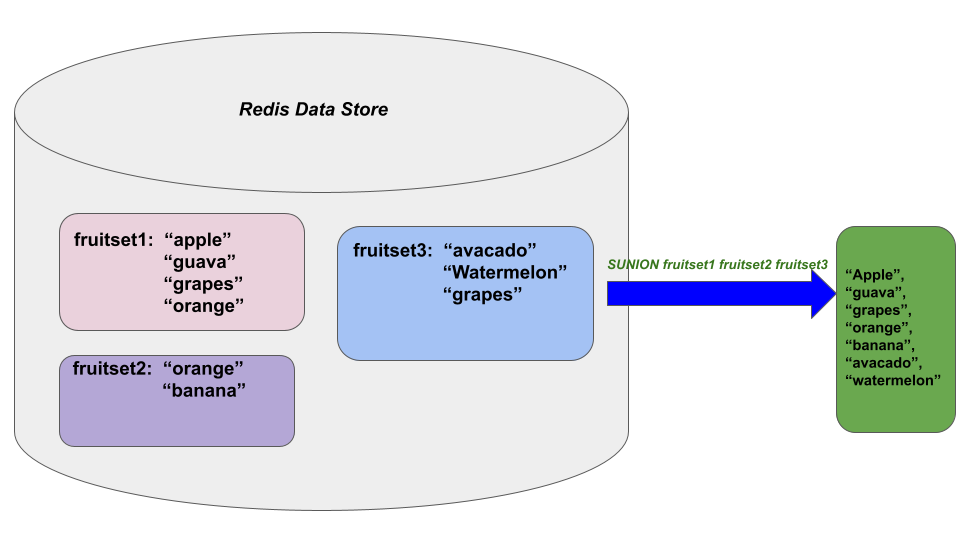
The following is the syntax of the SUNION command.

| SUNION set\_key [set\_key ...] |
| --- |

***set\_key:*** The key associated with the set.

The SUNION command accepts more than one set. If you have specified a single set in the command, the return would be all the elements belonging to the specified set.

This command returns another set that contains all the distinct members resulting from the union of given sets as shown in the following figure.



## Use Case: Get All the Visitors to Company Websites

Let’s assume that a company maintains 3 websites for different marketing purposes. Each website has a big number of registered users. During the Christmas season, the company is planning to send gift vouchers to all the active users who visited their 3 websites. The company keeps track of monthly visitors in a redis data store. Since they need to maintain distinct users who visit per month, they are using Redis sets here.

If user A visits the website in the early of this month will be added to the set. But the following visits in the same month for user A will not add a new record to the database.

Let’s create three sets with some dummy user IDs using the SADD command as follows.

The first set *site1:visitors* is used to hold the user's visit to site 1.

| sadd site1:visitors "Jack" "Harry\_12" "Mary4" "Saumya12" "Ricky\_martin" |
| --- |

Next, the *site2:visitors* set will be created as follows.

| sadd site2:visitors "Mary4" "Linkon" |
| --- |

Finally, the set *site3:visitors* which holds the visitors of site 3.

| sadd site3:visitors "Linkon" "Deva" "Martin" "John12" |
| --- |

Let’s inspect each set and verify all the members have been stored properly. We will be using the SMEMBERS command to fetch members per each set as follows.

| smembers site1:visitors |
| --- |

| smembers site2:visitors |
| --- |

| smembers site3:visitors |
| --- |

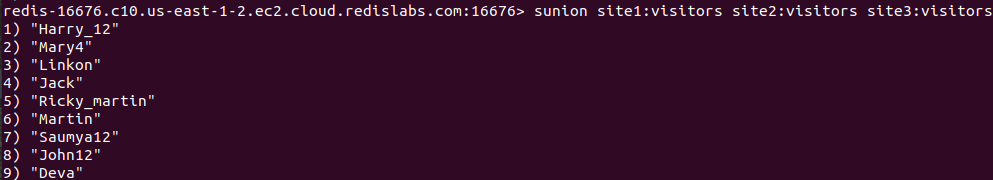
Output:



Now the company requires to retrieve all the distinguished visitors of all three sites. This is the union of three sets that we have created just before.

| sunion site1:visitors site2:visitors site3:visitors |
| --- |

Output:



As expected, the resulting list contains all the distinct members from all three sets: site1:visitors, site2:visitors, and site3:visitors.

Let’s specify a non-existing set in the command and see the output.

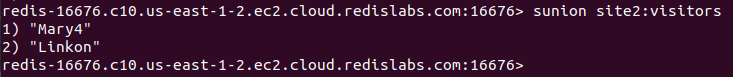
| sunion site1:visitors site2:visitors site3:visitors site4 |
| --- |

In the above command, *site4* is a non-existing set but there is no difference in the resulting collection of members. Because *site4* has been considered as an empty set by the SUNION command.

We can even specify a single set as the set argument. In that case, the union will contain all the elements in the specified set.

| sunion site2:visitors |
| --- |

Output:



As expected, the resulting set contains all the elements of *the site2:visitors* set. Overall, the SUNION command is very useful in taking the union of multiple sets which consumes a considerable amount of time when performed manually.

# Conclusion

To conclude, the SUNION command can be used to retrieve the union of given sets. This command operates on linear time complexity and has a very simple syntax where it accepts only the keys of the sets that you need to get the union of. As highlighted in the above sections, the SUNION command returns a list of members resulting from the union of the specified sets. Finally, the non-existing keys that are specified in the command will be considered empty sets.