**Title:** Redis ZINTERSTORE

***Excerpt:*** The ZINTERSTORE command is used to compute the set intersection for multiple sets specified. It stores the resulting intersection in a new sorted set. The final score value per member is calculated by summing the scores of each member across the source sets that they belong to. Also, the command supports specifying a multiplication factor for scores of the members in an intersection.

**Permalink:** redis-zinterstore

**Category:** Redis

# 

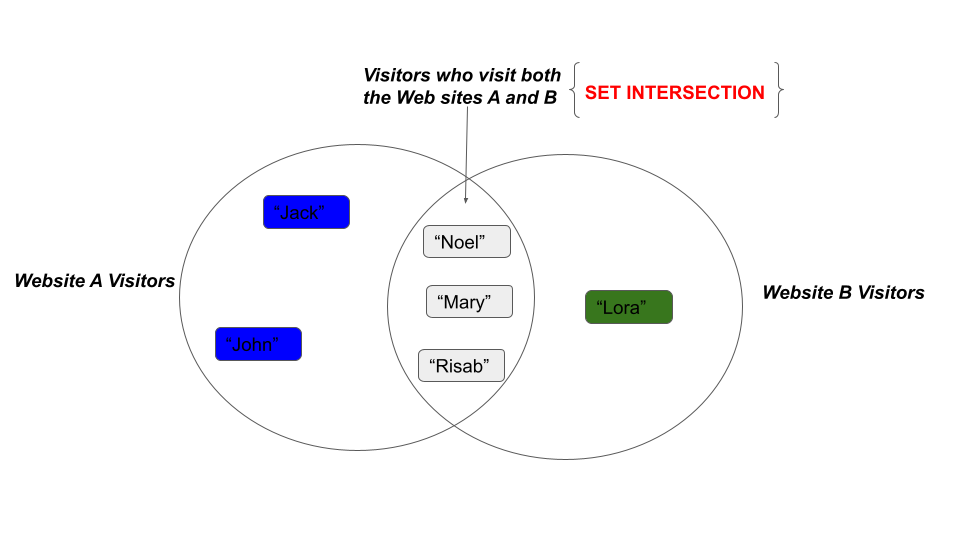
# 

# 

# Set Intersection

A set is a collection of elements such as numbers, letters, or real-world objects. Each of these members is distinct or unique for a given set. They alone can’t do many things. Hence, there are requirements to make relationships among two or more sets to generate meaningful insights. As we all know, numbers have fundamental operations such as addition, subtraction, multiplication, and division. In the same way, sets come with four main operations: union, intersection, difference, and complement.

In this guide, we will be focusing on the Redis command which operates on sorted sets to compute the intersection of two or more of them. Hence, this section explains the set intersection operation. As the name suggests, the set intersection operation computes the set of common elements belonging to a given list of sets.



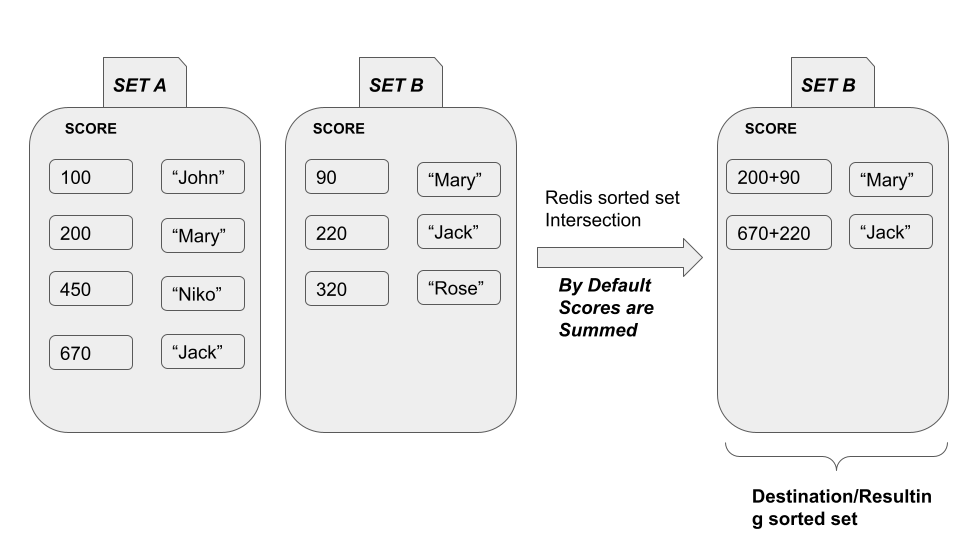
Above is a VENN diagram representation of two sets with an intersection. There are three members who visit both sites A and B. If we take the site A and B visitors as set A and set B, then the mentioned three members are called the set intersection of set A and set B.

Redis supports the sorted set data structure out of the box with general-purpose operations to add, remove and query the elements. Furthermore, Redis supports more advanced operations on sorted sets like set intersections. The following section describes the ZINTERSTORE command which helps in computing set intersection in Redis.

# Redis ZINTERSTORE Command

The ZINTERSTORE command operates on two or more sorted sets to compute the intersection of those. This command creates a new sorted set from the intersection of the specified sets.

Since, the Redis sorted set elements are associated with score values, each of these scores is summed per common element and stored in the destination set as shown in the following illustration.



## Syntax:

The following is the basic syntax of the ZINTERSTORE command.

| ZINTERSTORE destination\_set number\_of\_sets set\_key [set\_key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM | MIN | MAX] |
| --- |

***destination\_set:*** The key of the sorted set that will hold the intersection of the specified sorted sets.

***number\_of\_sets:*** The number of sorted sets that the set intersection will be computed against.

***set\_key:*** The key or unique identifier of the sorted set.

***WEIGHTS:*** The multiplication factor for each element’s score in the source sets.

***AGGREGATE:*** This option specifies a way to aggregate the resulting scores per element in the intersection.

By default, it takes the SUM of the scores per element among given source sets. It is possible to specify the minimum or maximum scores per element across the source sets that it belongs to.

Both the WEIGHTS and AGGREGATE arguments are optional to the ZINTERSTORE command.

The ZINTERSTORE command returns an integer value which is the number of members in the destination sorted set at *destination\_set*.

## Use Case - Inspect the Common Visitors Across Multiple Websites with their Visitor Counts.

Let’s assume a scenario where we got two websites A and B. To get an overall picture of the site visitors, we need to query the users who are visiting both A and B sites. Furthermore, we have a requirement to count the number of visits by each member.

Let’s create two sets setA and setB as shown in the following.

| zadd setA 600 "John" 150 "Mary" 300 "Nick" zadd setB 300 "Mary" 100 "Nick" 760 "Doe" |
| --- |

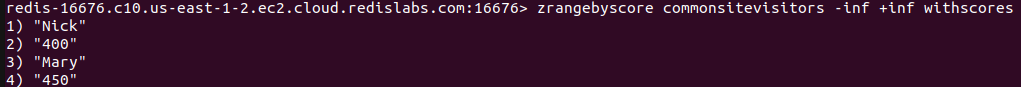
We can use the command ZINTERSCORE to find out the intersection of the setA and setB. Ideally the “Mary” and “Nick” should be the intersection of the above two sets.

| zinterstore commonsitevisitors 2 setA setB |
| --- |

In this example, we have used the *commonsitevisitors* as the key of the destination sorted set. It is mandatory to specify the number of sets that we use to compute the intersection. In this case, it is 2.



The returned value is 2 which means two members should be stored in the destination sorted set. Let’s inspect the resulting sorted set *commonsitevisitors* using the ZRANGEBYSCORE command.

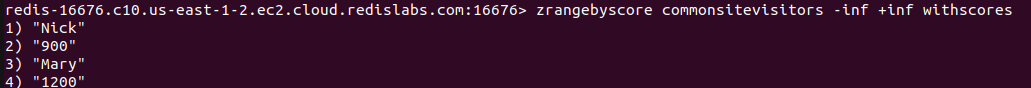


As expected, the “Nick” and “Mary” members are in the resulting sorted set with the summed score values. In this example, the member “Nick” has 300 and 100 scores in setA and setB respectively. Hence, the intersection of these two sets has summed the relevant score values for “Nick”. The same has happened with the member “Mary”.

Let’s use the multiplication factor 2 and 3 for setA and setB respectively.

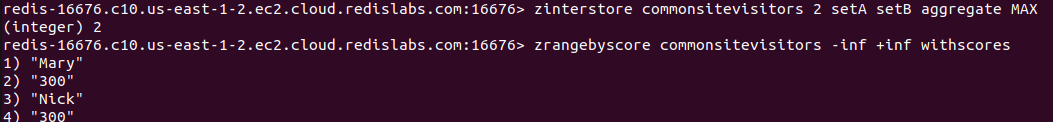
| zinterstore commonsitevisitors 2 setA setB weights 2 3 |
| --- |

The score of "Nick" will be calculated by multiplying 300 and 100 by 2 and 3 respectively and summing the results. Hence, the final score should be 900. The same procedure has been followed by the ZINTERSTORE command for the other member as well.



By default, the scores are aggregated by summing them, but the other options are available too. We can use the MIN and MAX arguments that will keep the minimum or maximum score per member in the resulting sorted set.

| zinterstore commonsitevisitors 2 setA setB aggregate MAX |
| --- |



As expected, the maximum score value for both members is 300 and it is kept in the destination sorted set.

# Conclusion

To summarize, the ZINTERSTORE command is used to compute the intersection for multiple sorted sets provided. It is capable of extracting the intersection and storing it in a new sorted set. As mentioned above, the scores per member across source sets will be summed by default. The minimum and maximum arguments can be passed to the command where the scores will be aggregated by minimum or maximum score in the source sets. At the same time, it is possible to specify a multiplication factor for scores of each element in the intersection set. Overall, the ZINTERSTORE command is reliable and fruitful in computing the set intersections.