**Title:** Redis ZSCAN

***Excerpt:*** The ZSCAN command is used to retrieve members and scores of a sorted set without blocking the client or server. It has optional arguments such as MATCH and COUNT to filter out the returned members based on a pattern match and the member's count.

**Permalink:** redis-zscan

**Category:** Redis

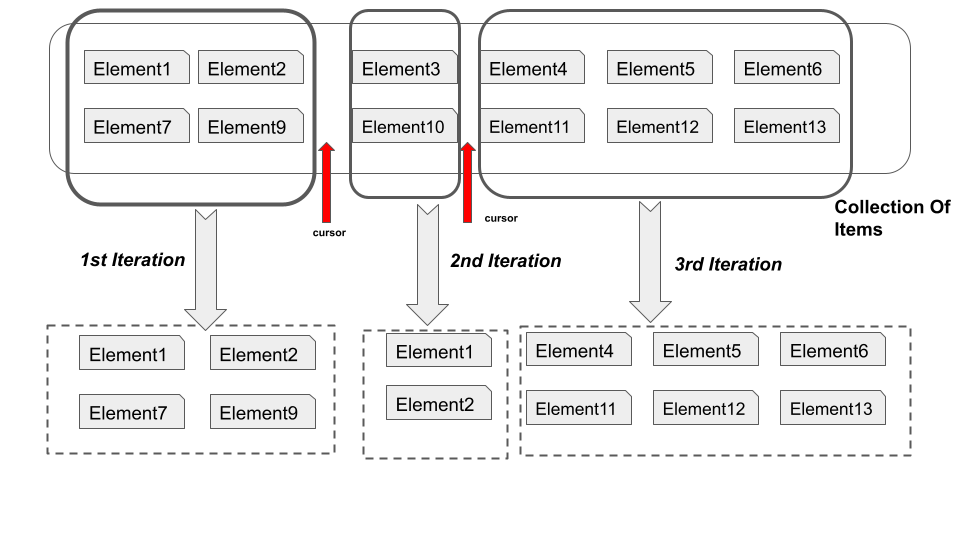
# 

# 

# 

# Iterate Over the Members of a Sorted Set

As you all know, Redis sorted sets are derived from regular sets where each member is ordered by its score value in ascending order. If two or more members hold the same score value, they will be ordered by the lexicographical order. Usually, the members and scores can be directly retrieved using the ZRANGE command. When you have a big sorted set with thousands of members, the ZRANGE command might block the server for a long time like SMEMBERS and KEYS commands which is a drawback. So, Redis offers a special command called ZSCAN which is derived from the SCAN command to iterate over the members of a sorted set. Since the ZSCAN command inherits from the SCAN command, almost all the behaviors are the same as the general purpose SCAN command.



As in the above figure, the SCAN command is a cursor-based iterator. Hence, it takes one or more iterations to provide all the items of a Redis collection. Since the ZSCAN command inherits from the parent SCAN command, behavior is the same. In this guide, the syntax and use cases of the ZSCAN command will be discussed in detail.

# The ZSCAN Command

The ZSCAN command is a cursor-based iterator that starts the iteration with the 0th cursor. Afterward, in each iteration, it will return zero or more sorted set members along with the next cursor that should be used as the cursor for the following command call. If the returned cursor is 0 after one or more iterations, it means the scanning process is over. All the sorted set members should have been returned at this point. This process is called a full iteration. As you could see, the ZSCAN command keeps its state only using a cursor which leads to a limited state awareness. Therefore, the following drawbacks are associated with the ZSCAN command.

* The same element may return in multiple iterations.
* If a member was not present at the start of the scanning process, there is a probability not to return that member during a full iteration.

In addition, there is no guarantee on the count of the returned members. In some cases, if the sorted set is very small, all the members might be returned in the very first iteration. Because, Redis uses a special single allocation packed encoding format to hold members until a maximum item count is reached. The ZSCAN command will only be able to return a cursor if the scanned data structure is represented as a hash table.

## Syntax

The ZSCAN command uses almost the same syntax as the SCAN command except that it accepts a sorted set key as the first argument. The command syntax with the allowed arguments is as follows.

| ZSCAN *sorted\_set\_key* *cursor* [MATCH *pattern*] [COUNT *member\_count*] |
| --- |

*sorted\_set\_key*:The key of the sorted set.

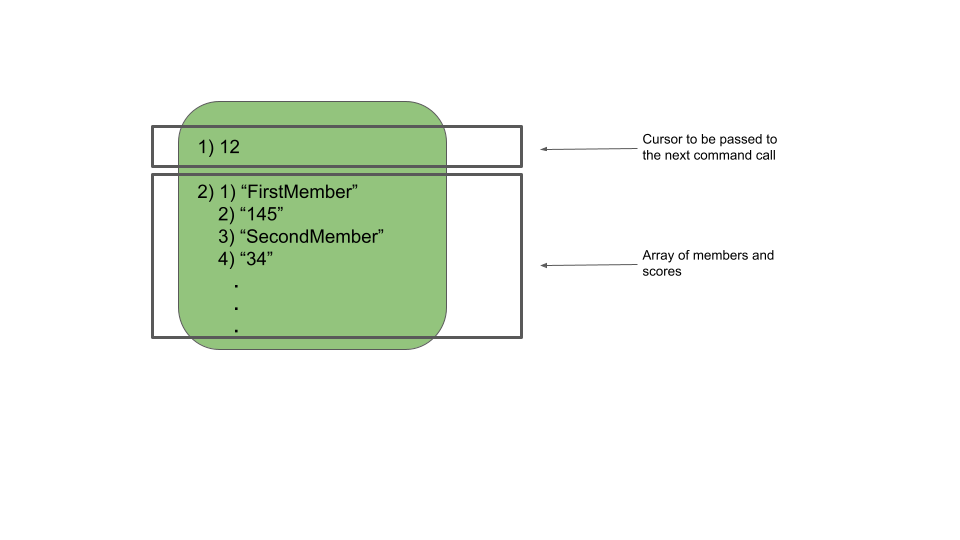
*Cursor*: The cursor value starts from 0 and ends at 0 if it is a full iteration.

The following arguments are optional.

MATCH: A pattern to match when retrieving elements in each iteration. Only the matched members will be returned.

COUNT: Approximate number of members to be returned in each iteration.

The returned result set per iteration contains a couple of elements. The first part is a 64-bit unsigned integer that represents the cursor to be passed into the next call. The next part is an array of members and associated scores.



## Use Case 01 - Retrieve All the Members and their Completed Missions of an Online Game

Let’s assume that an online game company maintains a leaderboard using the Redis sorted set. Since massive users are actively playing the game, they need a way to retrieve each player and their associated score which is the number of completed missions. It is a must to perform the retrieval without blocking the server. So, the recommendation is to use the ZSCAN command as follows.

First we will be creating a sorted set with some players and the completed number of missions.

| zadd LeaderBoard 12 Player6:John 4 Player2:Mary 22 Player1:Patel 15 Player:11 23 Player5:Ann 30 Player7:Harsh 23 Player12:abby 2 Player13:Nicky 6 Player9:Jeremy 7 Player45:Kina |
| --- |

Now we can iterate over the members of the sorted set as follows.

| zscan LeaderBoard 0 |
| --- |

Output:



The cursor value is 0 in the returned result set which means all the members have been returned at the end of the first iteration. In this case, because the number of members is small, Redis represents these members using a single-allocation packed encoding. Therefore, until a maximum pack size or member count is reached, the command will return all the members in the sorted set. This is called a Full iteration. Because at the end of the first iteration, we received all ten members and their scores. If we had hundreds of members, it would be represented as a hash table in memory. So, it would take several iterations to return all the members.

The COUNT parameter can be used to limit the number of members returned in an iteration. By default, this argument is set to 10. If the sorted set consists of hundreds of members, it is represented by a hash table in the memory. So, the number of returned members will be around ten per iteration. The value of the COUNT argument will be ignored if the sorted set is too small.

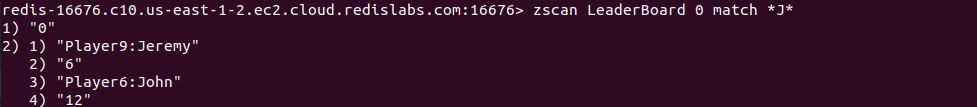
## Use Case 02 - Fetch the Players Whose Name Starts with Letter “J”

The ZSCAN command can be used to filter out the returned members based on a pattern match. In that case, the MATCH argument has to be specified.

Let’s use the same example from the previous use case. The requirement is to fetch players whose name starts with the letter “J”. It is just to implement the next cool feature related to the game. The MATCH argument can be specified as follows.

| zscan LeaderBoard 0 match \*J\* |
| --- |

This should ideally return two members whose names are Jeremy and John.



# Conclusion

In summary, the ZSCAN command is used to iterate over the members and scores of a Redis sorted set. This command behaves the same as the SCAN command except that the ZSCAN command accepts the set key as the first argument. As discussed in the use cases, the ZSCAN command can be used in different ways by specifying the MATCH and COUNT arguments where you can retrieve members and associated scores that match a specific pattern and limit the returned member count per iteration. Overall, the ZSCAN command can be useful when retrieving members of a sorted set without blocking the server or client.