Redis streams in production

Stream capping strategies

Documentation at redis.io:

EXPIRE command

XADD command

XLEN command

XRANGE command

XTRIM command

Enhancements in Redis 6.2: The MINID Trimming Strategy

In Redis 6.2, a new stream trimming strategy was added. It is called **MINID**, and allows us to trim a stream such that entries whose ID is lower than one provided are removed.
When combined with the default timestamp entry ID strategy, this gives us the ability to trim a stream to a given point in time.
In common with the **MAXLEN** trimming strategy, **MINID** can be used with both the **XADD** and **XTRIM** commands, as well as with the **~** modifier for approximate trimming.
Let's explore the **MINID** trimming strategy using a stream where we've added a new entry every 10 minutes for a period of 3 hours between midnight and 2:50am UTC on January 1st 2025.

First we'll create the stream, giving each entry a single name/value pair in its payload. We store the date/time in **YYYY-MM-DD HH:MM:SS** format so that we can easily see which time the entry ID timestamp represents:

```
127.0.0.1:6379> DEL demostream

(integer) 0

127.0.0.1:6379> XADD demostream 1735689600000-0 dt "2025-01-01 00:00:00"

"1735689600000-0"

127.0.0.1:6379> XADD demostream 1735690200000-0 dt "2025-01-01 00:10:00"

"1735690200000-0"

127.0.0.1:6379> XADD demostream 1735690800000-0 dt "2025-01-01 00:20:00"

"1735690800000-0"

127.0.0.1:6379> XADD demostream 1735691400000-0 dt "2025-01-01 00:30:00"

"1735691400000-0"

127.0.0.1:6379> XADD demostream 1735692000000-0 dt "2025-01-01 00:40:00"

"1735692000000-0"

127.0.0.1:6379> XADD demostream 1735692600000-0 dt "2025-01-01 00:50:00"

"1735692600000-0"

127.0.0.1:6379> XADD demostream 1735693200000-0 dt "2025-01-01 01:00:00"
```

```
"1735693200000-0"

127.0.0.1:6379> XADD demostream 1735693800000-0 dt "2025-01-01 01:10:00"

"1735693800000-0"

127.0.0.1:6379> XADD demostream 1735694400000-0 dt "2025-01-01 01:20:00"

"1735694400000-0"

127.0.0.1:6379> XADD demostream 1735695000000-0 dt "2025-01-01 01:30:00"

"1735695000000-0"

127.0.0.1:6379> XADD demostream 1735695600000-0 dt "2025-01-01 01:40:00"

"1735695600000-0"

127.0.0.1:6379> XADD demostream 1735696200000-0 dt "2025-01-01 01:50:00"

"1735696200000-0"

127.0.0.1:6379> XADD demostream 1735696800000-0 dt "2025-01-01 02:00:00"

"1735696800000-0"

127.0.0.1:6379> XADD demostream 1735697400000-0 dt "2025-01-01 02:10:00"

"1735697400000-0"

127.0.0.1:6379> XADD demostream 1735698000000-0 dt "2025-01-01 02:20:00"

"1735698000000-0"

127.0.0.1:6379> XADD demostream 1735698600000-0 dt "2025-01-01 02:30:00"

"1735698600000-0"

127.0.0.1:6379> XADD demostream 1735699200000-0 dt "2025-01-01 02:40:00"

"1735699200000-0"

127.0.0.1:6379> XADD demostream 1735699800000-0 dt "2025-01-01 02:50:00"

"1735699800000-0"
```

Having created the stream, we can run some checks to make sure that it is the expected length and that the first and last entries represent the time period we are modeling:

```
127.0.0.1:6379> XLEN demostream

 (integer) 18

 127.0.0.1:6379> XRANGE demostream - + COUNT 1

 1) 1) "1735689600000-0"

   2) 1) "dt"

      2) "2025-01-01 00:00:00"

 127.0.0.1:6379> XREVRANGE demostream + - COUNT 1

 1) 1) "1735699800000-0"

    2) 1) "dt"

      2) "2025-01-01 02:50:00"
```

Time Based Trimming with XADD

Imagine that we want to trim the stream so that only the first hour's entries are removed. Using the **MINID** strategy, we can compute an appropriate timestamp value (in our application code) and use that as an argument to either **XADD** or **XTRIM**.
Here we're trimming the stream to remove entries older than 01:00:00 on Jan 1st 2025 (**1735693200000**) while adding a new entry for 03:00:00 on Jan 1st 2025 (**1735700400000**) at the same time:

```
127.0.0.1:6379> XADD demostream MINID 1735693200000 1735700400000-0 dt "2025-01-01 03:00:00"

 "1735700400000-0"
```

Note that we don't have to add the sequence number to the value for **MINID**. If not supplied, a sequence number of **-0** is assumed.
Let's see how running this single command has affected the state of the stream:

```
127.0.0.1:6379> XLEN demostream

 (integer) 13

 127.0.0.1:6379> XRANGE demostream - + COUNT 1
```

```
1) 1) "1735693200000-0"
   2) 1) "dt"
      2) "2025-01-01 01:00:00"
127.0.0.1:6379> XREVRANGE demostream + - COUNT 1
1) 1) "1735700400000-0"
   2) 1) "dt"
      2) "2025-01-01 03:00:00"
```

Before running the command there were 18 entries in the stream. The **XADD** command added one more, and also atomically trimmed the 6 entries whose ID was less than **1735693200000** (those entries covering the time period from 00:00 to 00:50). This leaves us with a stream containing 13 entries that span the time period 01:00 to 03:00 on Jan 1st 2025.

Time Based Trimming with XTRIM

We can also trim the stream without adding a new entry to it. We do this with the **XTRIM** command.
Let's use **XTRIM** to remove entries older than 02:00 on Jan 1st 2025 UTC from our stream. As before, we need to calculate the millisecond timestamp for that date and time. It is **1735696800000**.
Next we call **XTRIM** using the **MINID** trimming strategy:

```
127.0.0.1:6379> XTRIM demostream MINID 1735696800000
(integer) 6
```

Redis responds with the number of entries that were trimmed from the stream: 6 in this case.
Note that, in common with **XADD**, we do not have to provide the sequence ID part of the minimum ID to trim to. **-0** is implied unless an explicit sequence ID is provided.
Let's see how running **XTRIM** has affected the state of the stream:

```
127.0.0.1:6379> XLEN demostream
(integer) 7
```

```
127.0.0.1:6379> XRANGE demostream - +
1) 1) "1735696800000-0"
  2) 1) "dt"
    2) "2025-01-01 02:00:00"
2) 1) "1735697400000-0"
  2) 1) "dt"
    2) "2025-01-01 02:10:00"
3) 1) "1735698000000-0"
  2) 1) "dt"
    2) "2025-01-01 02:20:00"
4) 1) "1735698600000-0"
  2) 1) "dt"
    2) "2025-01-01 02:30:00"
5) 1) "1735699200000-0"
  2) 1) "dt"
    2) "2025-01-01 02:40:00"
6) 1) "1735699800000-0"
  2) 1) "dt"
    2) "2025-01-01 02:50:00"
7) 1) "1735700400000-0"
  2) 1) "dt"
    2) "2025-01-01 03:00:00"
```

The stream was trimmed to contain only the 7 most recent entries. These cover the time period from 02:00 to 03:00.

Further Enhancements in Redis 6.2: The LIMIT Clause for Approximate Trimming

Redis 6.2 also added a **LIMIT** clause, which gives you a finer level of control over the time that stream trimming can take. This works with both the **XADD** and **XTRIM** commands in conjunction with the **~** approximate trimming modifier.
Imagine that we want our stream length to trend towards a 5000 entry cap, but that each time we add something new we only want to trim at most 1000 entries from the stream at a time. This helps us balance the time taken to run the trimming command with the need for other clients to have their commands executed by Redis.
First, let's create a very basic stream with 10000 entries in it:

```
127.0.0.1:6379> DEL demostream

 (integer) 0

 127.0.0.1:6379> 10000 XADD demostream * hello world

 "1680698899048-0"

 "1680698899050-0"

 "1680698899051-0"

 ...

 127.0.0.1:6379> XLEN demostream

 (integer) 10000
```

Now, let's run an **XTRIM** command that trims the stream towards an approximate 5000 entries, removing 1000 at a time at most:

```
127.0.0.1:6379> XTRIM demostream MAXLEN ~ 5000 LIMIT 1000

 (integer) 1000

 127.0.0.1:6379> XLEN demostream

 (integer) 9000
```

When we run this command again, we see another 1000 gone:

```
127.0.0.1:6379> XTRIM demostream MAXLEN ~ 5000 LIMIT 1000

(integer) 1000

127.0.0.1:6379> XLEN demostream

(integer) 8000
```

Finally, let's run the command without the **LIMIT** modifier:

```
127.0.0.1:6379> XTRIM demostream MAXLEN ~ 5000

(integer) 3000

127.0.0.1:6379> XLEN demostream

(integer) 5000
```

This trims more entries in a single command (5000), but that command takes longer to execute.
Use the **LIMIT** modifier if you want to have fine control over the throughput in Redis when trimming your stream to an approximate number with either **XTRIM** or **XADD**.
You can also use the **LIMIT** clause when using the **MINID** trimming strategy.

Additional Resources

For more information on the **MINID** trimming strategy and **LIMIT** clause, check out the XTRIM command page on redis.io.

← Previous

Next →

Modules

Course overview

Lesson

Course overview

☐ Lesson

Environment setup

Advanced consumer group management

☐ Lesson

Managing pending messages

☐ Assessment

Quiz 1 | Redis streams in production

☐ Lesson

Consumer recovery & poison-pill messages

☐ Assessment

Quiz 2 | Redis streams in production

☐ Lesson

The XAUTOCLAIM command

Performance and memory management

☐ Lesson

Redis

Cloud

Software

Pricing

Support