Garbage collection

This page describes how garbage collection works in Cloud Bigtable and covers the following topics:

- Types of garbage collection
- Default garbage-collection settings
- When data is removed
- Changes to garbage-collection policies for replicated tables

Overview of garbage collection

Garbage collection is the automatic, ongoing process of removing expired and obsolete data from Cloud Bigtable tables. A **garbage-collection policy** is a set of rules you create that state when data in a specific column family is no longer needed.

In this documentation, the term **remove** refers to data that is removed from tables in the course of routine gar ion. The term **delete** refers to data that is intentionally deleted from a table by an application or user.

Garbage collection is a built-in, asynchronous background process. **It can take up to a week before data that is eligible for garbage collection is actually removed.** Garbage collection occurs on a fixed schedule that does not vary based on how much data needs to be removed. Until the data is removed, it will appear in read results. You can filter your reads to exclude this data.

int: You do not need to filter values from reads if the values have been *deleted* by you or your application. Wh a value, it immediately stops appearing in read results. On the other hand, if you want to exclude values that a *d or removed as part of garbage collection*, always apply a filter to your read requests that excludes the same v r garbage-collection rules. See When data is removed (#data-removed) for further details.

The benefits of garbage-collection policies include the following:

- **Minimize row size** - You always want to prevent rows from growing indefinitely. Large rows negatively affect performance. Ideally, you should never let a row grow beyond 100 MB in size, and the limit is 256 MB. If you don't need to keep old data, or old versions of your current data, using garbage collection can help you minimize the size of each row.
- **Keep costs down** - Garbage collection ensures that you don't pay to store data that is no longer required or used. You are charged for storage of expired or obsolete data until compaction occurs (/bigtable/docs/overview#compactions) and garbage-collected data is removed. This process typically takes a few days but might take up to a week.

You can set garbage-collection policies either programmatically or with the `cbt` tool (/bigtable/docs/cbt-overview). Garbage-collection policies are set at the column-family level.

Each column family in a table has its own garbage-collection policy. The garbage collection process looks up the current garbage-collection policy for each column family, then removes data according to the rules in the policy.

Timestamps

In Cloud Bigtable, the intersection of a row and a column can have multiple cells, which are different versions of the value at that intersection. Each cell has a timestamp. A timestamp is the number of microseconds since the Unix epoch (https://wikipedia.org/wiki/Unix_time), `1970-01-01 00:00:00 UTC`. You can use default timestamps or set them when you send write requests (/bigtable/docs/writes).

The timestamp property of a cell can be a "real" timestamp, reflecting the actual time the value for the cell is written, or it can be an "artificial" timestamp. Artificial timestamps include sequential numbers, zeroes, or timestamp-formatted values that are not the actual time the cell was written. Before you use artificial timestamps, review the use cases for artificial timestamps, including the risks of using them:

- Simulating cell-level TTL (/bigtable/docs/gc-cell-level)
- Storing sequential numbers in timestamps (/bigtable/docs/gc-sequence-numbers)
- Keeping only the latest value (/bigtable/docs/gc-latest-value)

Types of garbage collection

This section describes the types of garbage collection available in Cloud Bigtable. Code samples for each type of garbage collection are at configuring garbage collection (/bigtable/docs/configuring-garbage-collection).

Expiring values (age-based)

You can set a garbage-collection rule based on the timestamp for each cell. For example, you might not want to keep any cells with timestamps more than 30 days before the current date and time. With this type of garbage-collection rule, you set the time to live (TTL) for data. Cloud Bigtable looks at each column family during garbage collection and removes any cells that have expired.

Number of versions

You can set a garbage-collection rule that explicitly states the maximum number of cells to keep for all columns in a column family.

For instance, if you want to keep only the latest username and email address for a customer, you can create a column family containing those two columns and set the maximum number of values to 1 for that column family.

In another case, you might want to keep the last 5 versions of a user's password hash to make sure they don't reuse the password, so you would set the maximum number of versions for the column family containing the password column to 5. When Cloud Bigtable looks at the column family during garbage collection, if a 6th cell has been written to the password column, the oldest cell is garbage-collected to keep the number of versions to 5.

Combinations of expiration and version-number rules

Two types of garbage-collection policies are available to combine these types of garbage-collection rules: **intersection** and **union**.

Intersection

An intersection garbage-collection policy will remove all data matching **all** of a set of given rules. For example, you might want to remove profiles older than 30 days but always keep at least one for each user. In this case, your intersection policy for the column family containing

the profile column would consist of a rule for an expiring value **and** a rule for the number of versions.

Union

A union garbage-collection policy will remove all data matching **any** of a set of given rules. For example, you might want to make sure you retain a maximum of 2 page-view records per user but only if they are less than 30 days old. In this case, your union policy would be set for an expiring value **or** a number of versions.

Default settings for garbage collection

There is no default TTL for a column family. The default number of versions for a column depends on how you create the column family that the column is in, as explained in the following sections.

HBase policy

If you create the column family with the HBase client for Java, the HBase shell, or another tool that uses the HBase client for Java, Cloud Bigtable retains only **the most recent version** of each value in the column family, unless you change the rule. This default setting is consistent with HBase.

All other client libraries or tools

If you create the column family with any other client library or tool, Cloud Bigtable retains an **infinite number** of versions of each value. This includes column families created with `gcloud` and the `cbt` tool. You must change the garbage-collection policy for the column family if you want to limit the number of versions.

When data is removed

Garbage collection is a continuous process in which Cloud Bigtable checks the rules for each column family and removes expired and obsolete data accordingly. In general, it can take up to

a week from the time that data matches the criteria in the rules for the data to actually be removed. You are not able to change the timing of garbage collection.

Garbage collection and deletes written to the table by your application are both executed during compaction, b ork differently and independently of each other. A key difference is that if your application deletes a piece of da equest for that data immediately afterward will return an empty response. On the other hand, when a piece of es eligible for garbage collection, you can continue to read it until the next compaction occurs.

Because it can take up to a week for data to be garbage-collected, you should never rely solely on garbage-collection policies to ensure that read requests return the desired data. **Always apply a filter to your read requests that excludes the same values as your garbage-collection rules.** You can filter by limiting the number of cells per column (/bigtable/docs/using-filters#cell-section) or by specifying a timestamp range (/bigtable/docs/using-filters#timestamp-range).

For example, let's say that a column-family's garbage-collection rule is set to keep only the 5 most recent versions of a profile, and 5 versions are already stored. After a new version of the profile is written, it might take up to a week for the oldest cell to be garbage-collected. Therefore, to avoid reading the 6th value, you should always filter out everything except the 5 most recent versions.

You are charged for storage of expired data until compaction occurs and the garbage-collected data is removed.

Garbage collection is retroactive: when a new garbage-collection policy is set, over the next few days it will be applied to all data in the table. If the new policy is more restrictive than the previous policy, old data will be removed as the background work happens, affecting data that was written before the policy change.

If you want to make sure data is being garbage-collected, you can query your table and compare the data with expected results. You can also monitor table size in the Google Cloud Console (/bigtable/docs/monitoring-instance#console-monitoring). A table that never gets smaller might reflect a garbage-collection policy that is not working as expected, but remember that garbage collection is executed on a delay.

Changes to garbage-collection policies for replicated tables

Cloud Bigtable allows you to modify or delete a policy for a column family at any time if the table is in a single-cluster instance. On the other hand, some restrictions apply to tables in replicated instances. These restrictions protect your data.

You can modify a column family's maximum number of versions in a replicated table. However, if you lower the number of versions for a column family, it could take up to a week for all replicated clusters to reflect the new, lower number. Therefore, you should always use filters when reading the data.

Cloud Bigtable will not allow you to increase the TTL for a column family in a replicated table (/bigtable/docs/replication-overview). To see why, consider a case where you want to change a column family's TTL from 30 days to 60 days. Age-based garbage collection can run separately in each cluster. As a result, at the time you change the policy, garbage collection might have removed a 31-day-old value in one copy of a table but not in another. Changing the garbage-collection policy in this situation can leave the copies out of sync for almost a month.

For the same reason, Cloud Bigtable will not allow you to delete an age-based garbage-collection policy for a column family in a replicated table.

What's next

- Explore strategies to simulate cell-level TTL (/bigtable/docs/gc-cell-level).
- Read about how timestamps that are sequential numbers (/bigtable/docs/gc-sequence-numbers) affect garbage collection.
- Learn how to only keep the most recent value of a column (/bigtable/docs/gc-latest-value).
- Learn more about storage pricing (/bigtable/pricing#storage).
- Look at garbage-collection code samples (/bigtable/docs/configuring-garbage-collection) in your preferred programming language.

Last updated 2020-08-03 UTC.