When reading data in Cloud Spanner in either a read-only transaction (/spanner/docs/transactions#read-only_transactions) or a single read call (/spanner/docs/reads), you can set a **timestamp bound**, which tells Cloud Spanner how to choose a timestamp at which to read the data.

Why set a timestamp bound? If your database is geographically distributed (that is, you created your Cloud Spanner instance using a multi-region instance configuration), and your application can tolerate some staleness when reading data, then you can get latency benefits from executing a stale read instead of a strong read. (Learn more about these read types in Reads (/spanner/docs/reads).)

The types of timestamp bound are:

- Strong (the default): read the latest data.
- Bounded staleness: read a version of the data that's no staler than a bound.
- Exact staleness: read the version of the data at an exact timestamp, e.g. a point in time in the past, though you can specify a timestamp for a time that hasn't passed yet. (If you specify a timestamp in the future, Cloud Spanner will wait for that timestamp before serving the read.)

Notes:

- Although reads using these timestamp bound modes are not part of a read-write transaction, they can block waiting for concurrent read-write transactions to commit. Bounded staleness reads attempt to pick a timestamp to avoid blocking, but may still block.
- Stale reads (i.e. using the bounded or exact staleness types) have the maximum performance benefit at longest staleness intervals. Use a minimum staleness of 10 seconds to get a benefit.

The Cloud Spanner timestamp bound types are explained in more detail below.

Cloud Spanner provides a bound type for strong reads. Strong reads are guaranteed to see the effects of all transactions that have committed before the start of the read. Furthermore, all rows yielded by a single read are consistent with each other - if any part of the read observes a transaction, all parts of the read see the transaction.

Strong reads are not repeatable: two consecutive strong read-only transactions might return inconsistent results if there are concurrent writes. If consistency across reads is required, the reads should be executed within the same transaction or at an exact read timestamp.

Cloud Spanner provides a bound type for bounded staleness. Bounded staleness modes allow Cloud Spanner to pick the read timestamp, subject to a user- provided staleness bound. Cloud Spanner chooses the newest timestamp within the staleness bound that allows execution of the reads at the closest available replica without blocking.

All rows yielded are consistent with each other - if any part of the read observes a transaction, all parts of the read see the transaction. Boundedly stale reads are not repeatable: two stale reads, even if they use the same staleness bound, can execute at different timestamps and thus return inconsistent results.

Bounded staleness reads are usually a little slower than comparable exact staleness reads.

If you're using bounded staleness with a read-only transaction, you can only use it with single-use read-only transact
th general-purpose read-only transactions.

Cloud Spanner provides a bound type for exact staleness. These timestamp bounds execute reads at a user-specified timestamp. Reads at a timestamp are guaranteed to see a consistent prefix of the global transaction history: they observe modifications done by all transactions with a commit timestamp less than or equal to the read timestamp, and observe none of the modifications done by transactions with a larger commit timestamp. They will block until all conflicting transactions that may be assigned commit timestamps less than or equal to the read timestamp have finished.

The timestamp can either be expressed as an absolute Cloud Spanner commit timestamp or a staleness relative to the current time.

These modes do not require a "negotiation phase" to pick a timestamp. As a result, they execute slightly faster than the equivalent boundedly stale concurrency modes. On the other hand, boundedly stale reads usually return fresher results.

Cloud Spanner continuously garbage collects deleted and overwritten data in the background to reclaim storage space. This process is known as **version GC**. Version GC reclaims versions after they are one hour old. Because of this, Cloud Spanner cannot perform reads at a read timestamp more than one hour in the past. This restriction also applies to in-progress reads and/or SQL queries whose timestamp become too old while executing. Reads and SQL queries with too-old read timestamps fail with the error `FAILED_PRECONDITION`.