Partitioned Data Manipulation Language (/spanner/docs/dml-syntax) (Partitioned DML) is designed for bulk updates and deletes:

- Periodic cleanup and garbage collection. Examples are deleting old rows or setting columns to `NULL`.
- Backfilling new columns with default values. An example is using an `UPDATE` statement to set a new column's value to `False` where it is currently `NULL`.

Cloud Spanner supports two execution modes for DML statements.

- DML is suitable for transaction processing.
- Partitioned DML enables large-scale, database-wide operations with minimal impact on concurrent transaction processing by partitioning the key space and running the statement over partitions in separate, smaller-scoped transactions.

The following table highlights some of the differences between the two execution modes.

DML	Partitioned DML
Rows that do not match the `WHERE` clause might be locked.	Only rows that match the `WHERE` clause are locked.
Transaction size limits apply.	Cloud Spanner handles the transaction limits and per-transaction concurrency limits.
Statements do not need to be idempotent.	A DML statement must be idempotent (#partitionable-idempotent) to guarantee consistent results.
A transaction can include multiple DML and SQL statements.	A partitioned transaction can include only one DML statement.
There are no restrictions on complexity of statements.	Statements must be fully partitionable (#partitionable-idempotent).
You create read-write transactions in your client code.	Cloud Spanner creates the transactions.

When a Partitioned DML statement runs, rows in one partition do not have access to rows in other partitions, and you cannot choose how Cloud Spanner creates the partitions. Partitioning ensures scalability, but it also means that Partitioned DML statements must be *fully partitionable*. That is, the Partitioned DML statement must be expressible as the union of a set of statements, where each statement accesses a single row of the table and each statement accesses no other tables. For example, a DML statement that accesses multiple tables or performs a self-join is not partitionable. If the DML statement is not partitionable, Cloud Spanner returns the error `BadUsage`.

These DML statements are fully partitionable, because each statement can be applied to a single row in the table:

This DML statement is not fully partitionable, because it accesses multiple tables:

Cloud Spanner might execute a Partitioned DML statement multiple times against some partitions due to network-level retries. As a result, a statement might be executed more than once against a row. The statement must therefore be *idempotent* to yield consistent results. A statement is idempotent if executing it multiple times against a single row leads to the same result.

This DML statement is idempotent:

This DML statement is not idempotent:

Cloud Spanner acquires a lock only if a row is a candidate for update or deletion. This behavior is different from DML execution (/spanner/docs/dml-tasks#locking), which might read-lock rows that do not match the `WHERE` clause.

Whether a DML statement is partitioned or not depends on the client library method that you choose for execution. Each client library provides separate methods for DML execution (/spanner/docs/dml-tasks#client-library-dml) and Partitioned DML execution (/spanner/docs/dml-tasks#client-library-partitioned).

You can execute only one Partitioned DML statement in a call to the client library method.

Cloud Spanner does not apply the Partitioned DML statements atomically across the entire table. Cloud Spanner does, however, apply Partitioned DML statements atomically across each partition.

Partitioned DML does not support commit or rollback. Cloud Spanner executes and applies the DML statement immediately.

- If you cancel the operation, Cloud Spanner cancels the executing partitions and doesn't start the remaining partitions. Cloud Spanner does not roll back any partitions that have already executed.
- If the execution of the statement causes an error, then execution stops across all partitions and Cloud Spanner returns that error for the entire operation. Some examples of errors are violations of data type constraints, violations of `UNIQUE INDEX`, and violations of `ON DELETE NO ACTION`. Depending on the point in time when the execution failed, the statement might have successfully run against some partitions, and might never have been run against other partitions.

If the Partitioned DML statement succeeds, then Cloud Spanner ran the statement at least once against each partition of the key range.

A Partitioned DML statement returns a lower bound on the number of modified rows. It might not be an exact count of the number of rows modified, because there is no guarantee that Cloud Spanner

counts all the modified rows.

Cloud Spanner creates the partitions and transactions that it needs to execute a Partitioned DML statement. Transaction limits or per-transaction concurrency limits apply, but Cloud Spanner attempts to keep the transactions within the limits.

Cloud Spanner allows a maximum of 20,000 concurrent Partitioned DML statements per database.

Cloud Spanner does not support some features for Partitioned DML:

- `INSERT` is not supported.
- Cloud Console: You can't execute Partitioned DML statements in the Cloud Console.
- Query plans and profiling: The `gcloud` command-line tool and the client libraries do not support query plans and profiling.

The following code example updates the `MarketingBudget` column of the `Albums` table.

The following code example deletes rows from the `Singers` table, based on the `SingerId` column.