How idempotence works

Idempotence is a critical consideration for data movement.

Idempotence means that if you execute an operation multiple times, the final result will not change after the initial execution.

Consider elevators; pushing the button for a particular floor will always send you to that floor no matter how many times you press it.

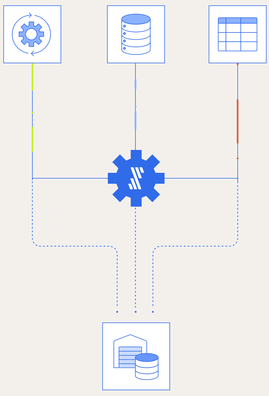
For data movement, idempotence ensures that if you apply the same data to a destination multiple times, you will get the same result. Without idempotence, you must sift through the data to identify duplicate records and develop a custom recovery procedure whenever a sync fails and must be repeated.

```
Mathematically, idempotence can be
expressed thusly:

f(f(x)) = f(x)

The absolute value function abs()
is idempotent

abs(abs(x)) = abs(x)
```



There are several places a sync may fail:

1. **At the source –** a data source may unexpectedly become unavailable, interrupting the sync. This can be the result of a network stoppage or a failure at the source itself.
2. **Within the pipeline –** Sometimes the pipeline itself just hiccups and stops working. There are many reasons a pipeline can fail:
   - Infrastructure failures, such as servers going down
   - Wrong or missing credentials
   - Resource limitations, such as memory leaks
   - Software bugs
3. **At the destination –** Failed queries to a data warehouse can interrupt syncs, as can upgrades and migrations. This may happen because of resource constraints, such as busy nodes and scheduled downtime.

Over a sufficiently long period of operation, a data pipeline will inevitably experience failures and introduce duplicate or conflicting records.

Walking through a simple example

One way to implement idempotence is to:
1) Track progress through batches of a sync using a cursor and
2) Use unique identifiers (i.e. primary keys) to identify and remove duplicate and conflicting records.

Suppose we have four micro batches of data to be synced to a data warehouse. The cursor, symbolized by a green checkmark, is basically a bookmark that records progress. In the below example, it indicates that the first batch has been completed. Here, the pipeline is partway through the second batch.

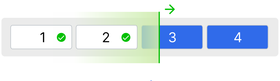
All records from the first batch ("foo," "bar," "baz," "quz" and "corge") and some from the second batch ("grault," "garply" and "waldo") have been loaded:

1	foo
	bar
	baz
	quz
	corge
2	grault
	garply
	waldo

Then, the sync is interrupted:

When the data pipeline resumes, it resumes from the last location indicated by the cursor, recognizing that the first batch has been completed but the second has not.

As the second batch is completed, the cursor is updated to the end of the second batch.



Records that were previously synced from the second batch (in this case, "grault," "garply" and "waldo") are reintroduced to the destination. Without idempotence, the records are duplicated (red rows).

1	foo
	bar
	baz
	quz
	corge
2	grault
	garply
	waldo
	grault
	garply
	waldo
	fred
	plugh

Duplicate records can pose serious problems for business intelligence and analytics:

1. **Misleading or erroneous insights** - you will not have correct rank orders, counts or sums of records. Duplication may also alter medians, averages, distributions and other summary metrics.
2. **Broken operations downstream** - any systems that rely on a one-to-one correspondence between records and identifiers may throw an error or produce the wrong behavior.
3. **Wasted storage space and computational bandwidth** - extra records don't add value yet still have to be accounted for in queries (and disk space).

We must identify every unique record using some sort of unique identifier and ensure there is no duplication so that the output of the process we illustrated above looks like so:

1	foo
	bar
	baz
	quz
	corge
2	grault
	garply
	waldo
	fred
	plugh

Idempotence is a key pillar of how Fivetran implements data movement.