Altering a keyspace

Changing a keyspace to use a different RF or strategy is a simple matter of using the `ALTER KEYSPACE` command. Let's assume that we have created a keyspace called `fenago_test`:

```
CREATE KEYSPACE fenago_test WITH replication = {
  'class': 'SimpleStrategy', 'replication_factor': '1'}
AND durable_writes = true;
```

As it is preferable to use `NetworkTopologyStrategy`, we can alter that easily:

```
ALTER KEYSPACE fenago_test  WITH replication = { 'class':'NetworkTopologyStrategy',
  'datacenter1': '1'};
```

If, at some point, we want to add our second data center, that command would look like this:

```
ALTER KEYSPACE fenago_test  WITH replication = { 'class': 'NetworkTopologyStrategy',
 'datacenter1': '1', 'datacenter2': '1'};
```

If we added more nodes to both data centers, and needed to increase the RF in each, we can simply run this:

```
ALTER KEYSPACE fenago_test  WITH replication = {'class': 'NetworkTopologyStrategy',
 'datacenter1': '3', 'datacenter2': '3'};
```

Note

Updating an RF on a keyspace does not automatically move data around. The data will need to be streamed via repair, rebuild, or bootstrap.

Dropping a keyspace

Removing a keyspace is a simple matter of using the `DROP KEYSPACE` command:

```
DROP KEYSPACE fenago_test;
```

Note

Dropping a keyspace does not actually remove the data from disk.

Altering a table

Tables can be changed with the `ALTER` statement, using it to add a column:

```
ALTER TABLE query_test ADD c6 TEXT;
```

Or to remove a column:

```
ALTER TABLE query_test DROP c5;
```

Note

Primary key definitions cannot be changed on a table. To accomplish this, the table must be recreated.

Table options can also be set or changed with the `ALTER` statement. For example, this statement updates the default TTL on a table to one day (in seconds):

```
ALTER TABLE query_test WITH default_time_to_live = 86400;
```

Truncating a table

To remove all data from a table, you can use the `TRUNCATE TABLE` command:

```
TRUNCATE TABLE query_test;
```

Dropping a table

Removing a table is a simple matter of using the `DROP TABLE` command:

```
DROP TABLE query_test;
```

Note

Try to avoid frequent drops or creates of a table with the same name. This process has proven to be problematic with Apache Cassandra in the past. If you need to recreate a table, it's always a good idea to `TRUNCATE` it before dropping it. It may be helpful to create a table with a version number on the end of it (`query_test_v2`) to prevent this problem from occurring.

Truncate versus drop

It's important to note that dropping a table is different from truncating it. A drop will remove the table from the schema definition, but the data will remain on-disk. With truncate, the data is removed, but the schema remains. Truncate is also the only way to clear all data from a table in a single command, as a CQL delete requires key parameters.