Executing a BATCH statement

One of the more controversial features of CQL is the `BATCH` statement. Essentially, this allows for write operations to be grouped together and applied at once, automatically. Therefore, if one of the statements should fail, they are all rolled back. A common application of this is for developers to write the same data to multiple query tables, keeping them in sync. `BATCH` statements in CQL can be applied as such:


```
BEGIN BATCH
 INSERT INTO security_logs_by_location (location_id,day,time_in,employee_id,mailstop)
 VALUES ('MPLS2',20180726,'2018-07-26 11:45:22.004','robp','M266');
 INSERT INTO security_logs_by_location_desc (location_id,day,time_in,employee_id,mailstop)
 VALUES ('MPLS2',20180726,'2018-07-26 11:45:22.004','robp','M266');
APPLY BATCH;
```

Note that statements in `BATCH` are not guaranteed to be executed in any particular order. All statements within a batch are assumed to have been applied at the same time, and so will bear the same write timestamps.

What makes this functionality controversial is that it bears a resemblance to a keyword in the RDBMS world, which behaves very differently. RDBMS developers are encouraged to apply several (sometimes thousands) of writes in a batch, to apply them all in one network trip, and gain some performance. With Cassandra, this approach is dangerous, because batching too many writes together can exacerbate a coordinator node and potentially cause it to crash.

Note

`BATCH` was just not designed to apply several updates to the same table. It was designed to apply one update to several (such as five or six) tables. This fundamental misunderstanding of the purpose of `BATCH` can potentially affect cluster availability.

The expiring cell

Apache Cassandra allows for data in cells to be expired. This is called setting a TTL. TTLs can be applied at write-time, or they can be enforced at the table level with a default value. To set a TTL on a row at write time, utilize the `USING TTL` clause:

```
 INSERT INTO query_test (pk1,pk2,ck3,ck4,c5)
VALUES ('f','g','c4','d6','e7')
USING TTL 86400;
```

Likewise, TTLs can also be applied with the `UPDATE` statement:

```
 UPDATE query_test
USING TTL 86400
SET c5='e7'
WHERE pk1='f' AND pk2='g' AND ck3='c4' AND ck4='d6';
```

TTLs represent the number of seconds elapsed since write-time.

Note

One day is 86,400 seconds.