Introduction to CQL

Paul O'Fallon

@paulofallon

Overview

Keyspaces, tables and basic data types

CRUD operations

Counters

Aggregate functions

A Brief History of Communicating with Cassandra

2008: Originally just a Thrift API

2011: CQL introduced in Cassandra 0.8

2012: CQL 3 introduced in Cassandra 1.1

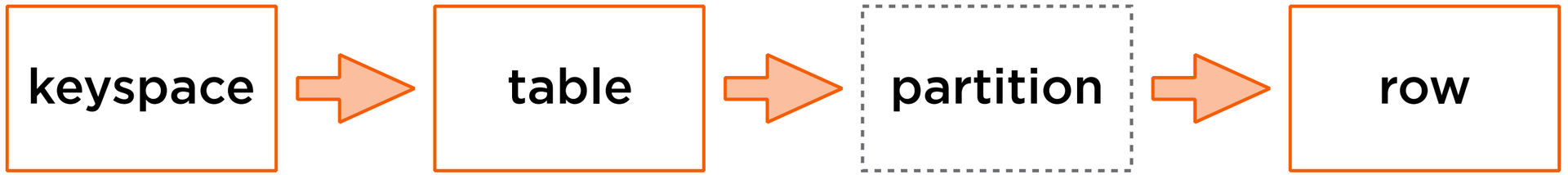
2013: CQL 3.1 introduced in Cassandra 2.0

2014: CQL 3.2 introduced in Cassandra 2.1

In this course: CQL 3.4.4

~~Column Family~~

~~Super Column Family~~



CQL on the Command Line

```
→ demos docker-compose exec n1 cqlsh --help
Usage: cqlsh.py [options] [host [port]]

CQL Shell for Apache Cassandra

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -C, --color           Always use color output
  --no-color            Never use color output
  --browser=BROWSER     The browser to use to display CQL help, where BROWSER
                        can be:
                        - one of the supported browsers in
                        https://docs.python.org/2/library/webbrowser.html.
                        - browser path followed by %s, example: /usr/bin
                        /google-chrome-stable %s
  --ssl                 Use SSL
  --no_compact          No Compact
  -u USERNAME, --username=USERNAME
                        Authenticate as user.
  -p PASSWORD, --password=PASSWORD
                        Authenticate using password.
  -k KEYSPACE, --keyspace=KEYSPACE
                        Authenticate to the given keyspace.
  -f FILE, --file=FILE  Execute commands from FILE, then exit
  --debug               Show additional debugging information
  --encoding=ENCODING   Specify a non-default encoding for output. (Default:
                        utf-8)
```

CQL on the Command Line

```
cqlsh> help

Documented shell commands:
==========================
CAPTURE  CLS          COPY  DESCRIBE  EXPAND  LOGIN    SERIAL  SOURCE   UNICODE
CLEAR    CONSISTENCY  DESC  EXIT      HELP    PAGING   SHOW    TRACING

CQL help topics:
================
AGGREGATES               CREATE_KEYSPACE            DROP_TRIGGER      TEXT
ALTER_KEYSPACE           CREATE_MATERIALIZED_VIEW   DROP_TYPE         TIME
ALTER_MATERIALIZED_VIEW  CREATE_ROLE                DROP_USER         TIMESTAMP
ALTER_TABLE              CREATE_TABLE               FUNCTIONS         TRUNCATE
ALTER_TYPE               CREATE_TRIGGER             GRANT             TYPES
ALTER_USER               CREATE_TYPE                INSERT            UPDATE
APPLY                    CREATE_USER                INSERT_JSON       USE
ASCII                    DATE                       INT               UUID
BATCH                    DELETE                     JSON
BEGIN                    DROP_AGGREGATE             KEYWORDS
BLOB                     DROP_COLUMNFAMILY          LIST_PERMISSIONS
BOOLEAN                  DROP_FUNCTION              LIST_ROLES
COUNTER                  DROP_INDEX                 LIST_USERS
CREATE_AGGREGATE         DROP_KEYSPACE              PERMISSIONS
CREATE_COLUMNFAMILY      DROP_MATERIALIZED_VIEW     REVOKE
CREATE_FUNCTION          DROP_ROLE                  SELECT
CREATE_INDEX             DROP_TABLE                 SELECT_JSON
```

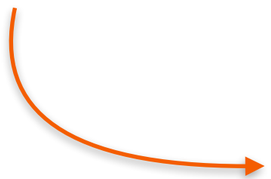
Keyspaces

Create

```
CREATE KEYSPACE pluralsight WITH REPLICATION = {
   'class': 'NetworkTopologyStrategy', 'DC1': 3
} AND DURABLE_WRITES = false;
```

Alter

```
ALTER KEYSPACE pluralsight WITH REPLICATION = {
   'class': 'SimpleStrategy', 'replication_factor': 3
} AND DURABLE_WRITES = true;
```



nodetool repair **required**

Drop

```
DROP KEYSPACE pluralsight;
```

Tables

Create

```
CREATE TABLE pluralsight.courses (id varchar PRIMARY KEY);
```

Alter

```
ALTER TABLE pluralsight.courses ADD name varchar;
ALTER TABLE pluralsight.courses DROP title;
```

Truncate

```
TRUNCATE pluralsight.courses;
```

Drop

```
DROP TABLE pluralsight.courses;
```

Table Properties

```
CREATE TABLE pluralsight.courses (id varchar PRIMARY KEY)
WITH comment='A table of courses';
```

- comment
- caching (keys, rows_per_partition)
- read_repair_chance
- dclocal_read_repair_chance
- default_time_to_live
- gc_grace_seconds
- bloom_filter_fp_chance
- compaction
- compression
- min/max_index_interval
- memtable_flush_period_in_ms
- populate_io_cache_on_flush
- speculative_retry

Basic Data Types in Cassandra

Numeric

bigint, decimal, double, float, int, varint

String

ascii, text, varchar

Date

timestamp, timeuuid

Other

boolean, uuid, inet, blob

Naming Your Keyspaces, Tables and Columns

- **No hyphens: 2015-stats ✗**
- **No spaces: 2015 stats ✗**
- **Double quotes required for initial digits: “2015stats”**
- **Mixed case is lowered unless in double quotes: “firstName”**

Primary Keys and Composite Partition Keys

```
CREATE TABLE pluralsight.courses (
  id varchar PRIMARY KEY,
  title varchar,
  author varchar
);
```

```
CREATE TABLE pluralsight.courses (
  id varchar,
  title varchar,
  author varchar,
  PRIMARY KEY ((id, author))
);
```

ONE ROW
PER PARTITION

(for now...)

Demo

Create and alter a courses table

Selecting Data

```
SELECT id, title FROM pluralsight.courses;

SELECT title, duration AS length FROM pluralsight.courses
WHERE id = 'cassandra-developers';

SELECT title, published FROM pluralsight.courses
WHERE id IN (‘cassandra-developers’,’nodejs-big-picture’);

SELECT * FROM pluralsight.courses LIMIT 100;
```

Inserting and Updating Data

Insert

```
INSERT INTO pluralsight.courses (id, author)
VALUES ('cassandra-developers','paul-ofallon');
```

UPSERT!

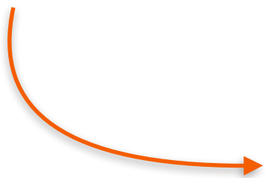
Update

```
UPDATE pluralsight.courses SET author = 'paul-ofallon'
WHERE id = 'cassandra-developers';

UPDATE pluralsight.courses SET author = 'paul-ofallon'
WHERE id in (‘cassandra-developers’,’nodejs-big-picture');
```

When Was the Data Written?

```
SELECT id, WRITETIME(author) FROM pluralsight.courses;
```



Unix time (e.g. 1430825689)

Deleting Data

Deleting a row

```
DELETE FROM pluralsight.courses
WHERE id = ‘cassandra-developers’;
```

Deleting a column

```
DELETE author FROM pluralsight.courses
WHERE id = ‘cassandra-developers’;

UPDATE pluralsight.courses SET author = null
WHERE id = ‘cassandra-developers’;

INSERT INTO pluralsight.courses (id, author)
VALUES (‘cassandra-developers’, null);
```

Expiring Data with TTLs

Set the TTL for a single column value

```
UPDATE pluralsight.users USING TTL 32400
SET reset_token = '1GRhEs1' WHERE id = 'john-doe';
```

Retrieve the TTL for a column value

```
SELECT TTL(reset_token) FROM pluralsight.users
WHERE id='john-doe';
```

Expiring Data with TTLs

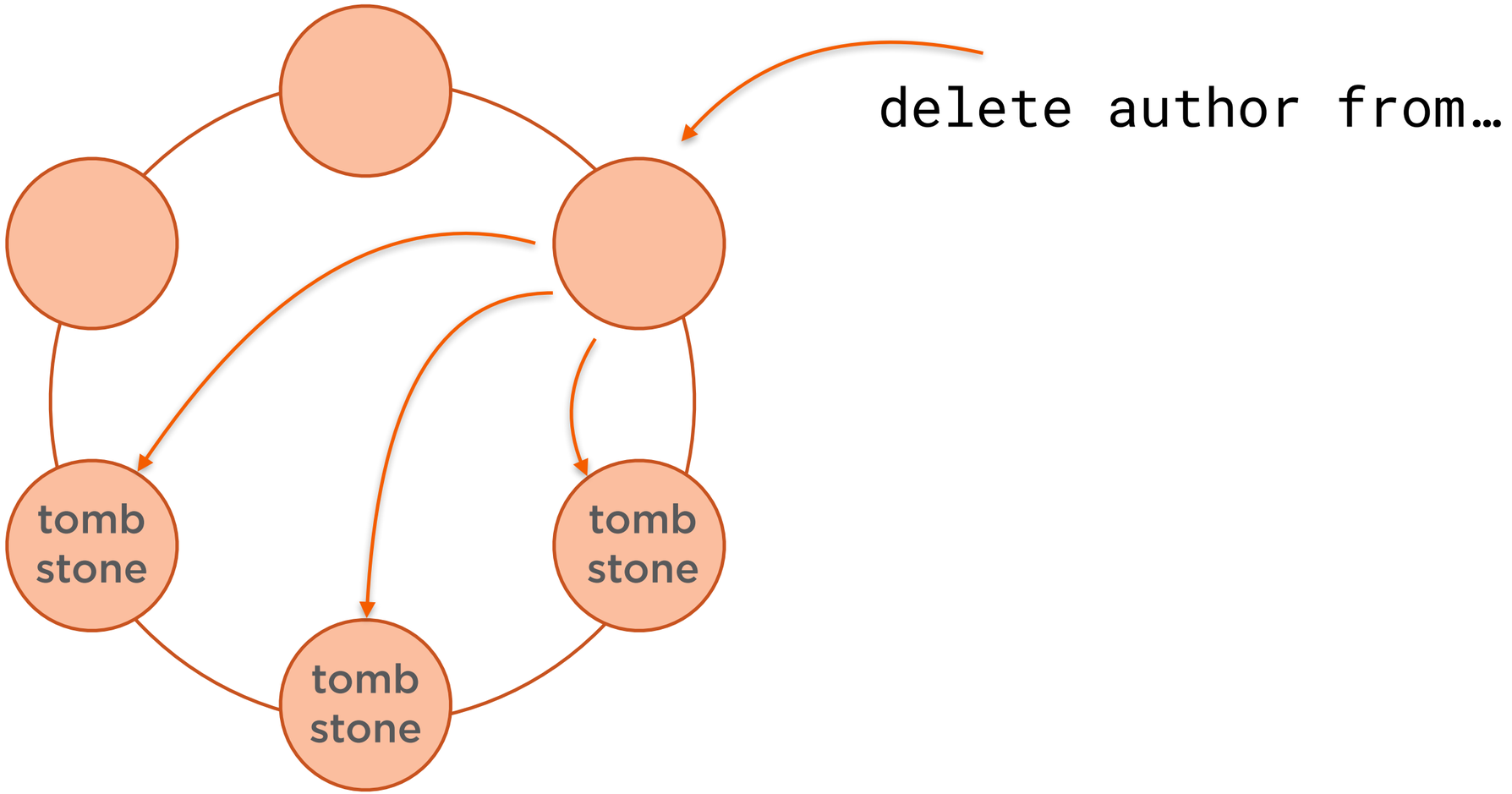
Set the TTL for an entire row

```
INSERT INTO pluralsight.reset_tokens (id, token)
VALUES ('john-doe','1GRhEs1') USING TTL 10800;
```

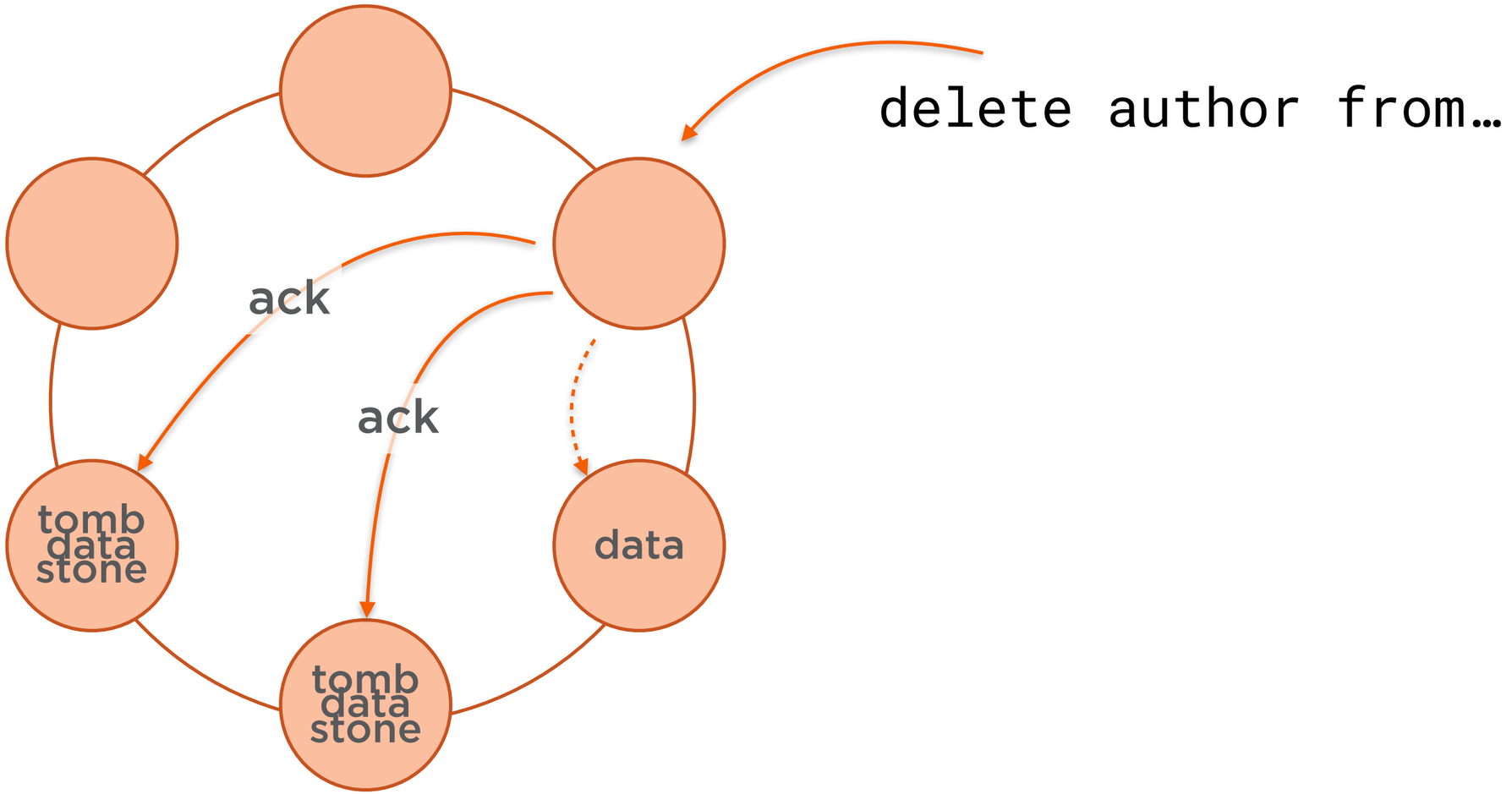
Set a table-wide, default TTL

```
CREATE TABLE reset_tokens (
   id varchar PRIMARY KEY,
   token varchar
) WITH default_time_to_live = 10800;
```

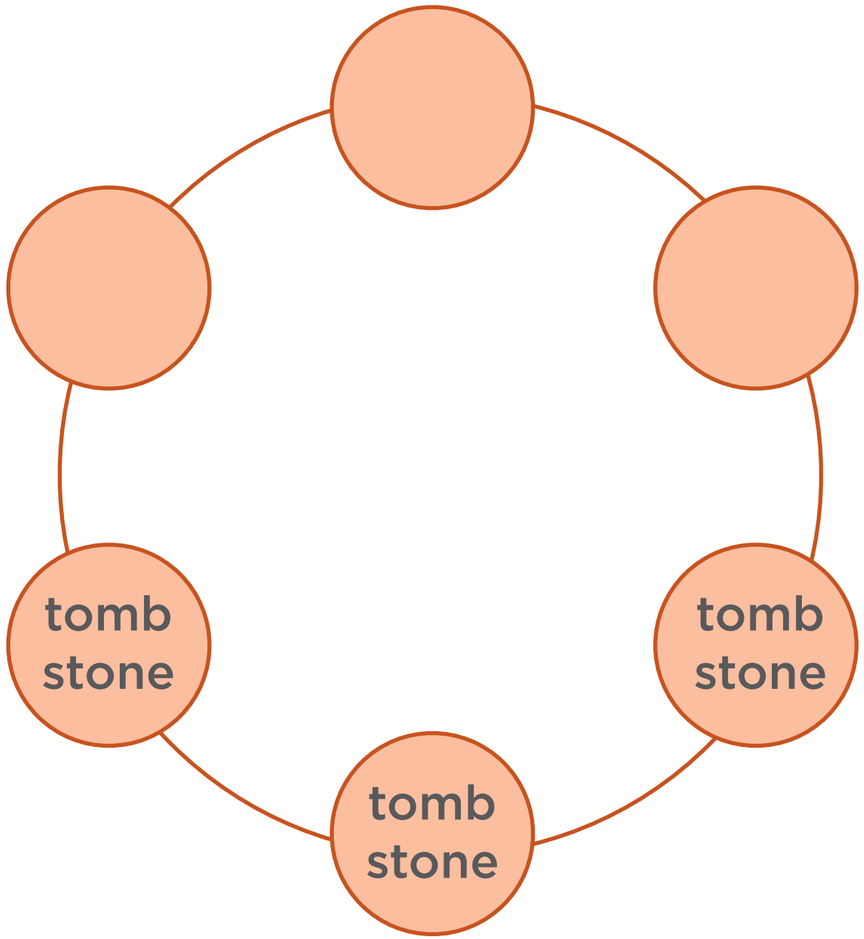
Tombstones



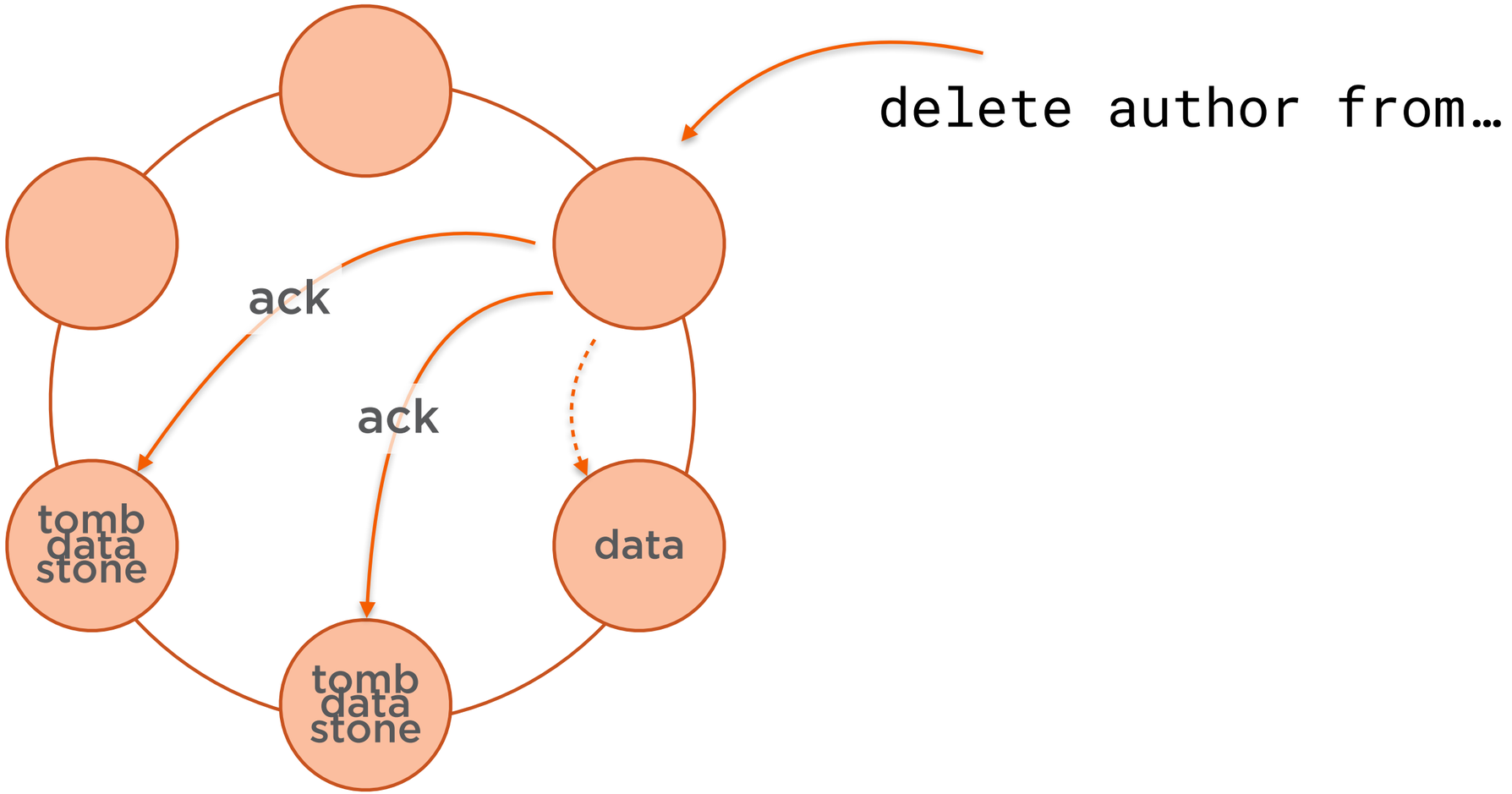
Tombstones


delete author from…
ack
ack
tomb
data
stone
data
tomb
data
stone

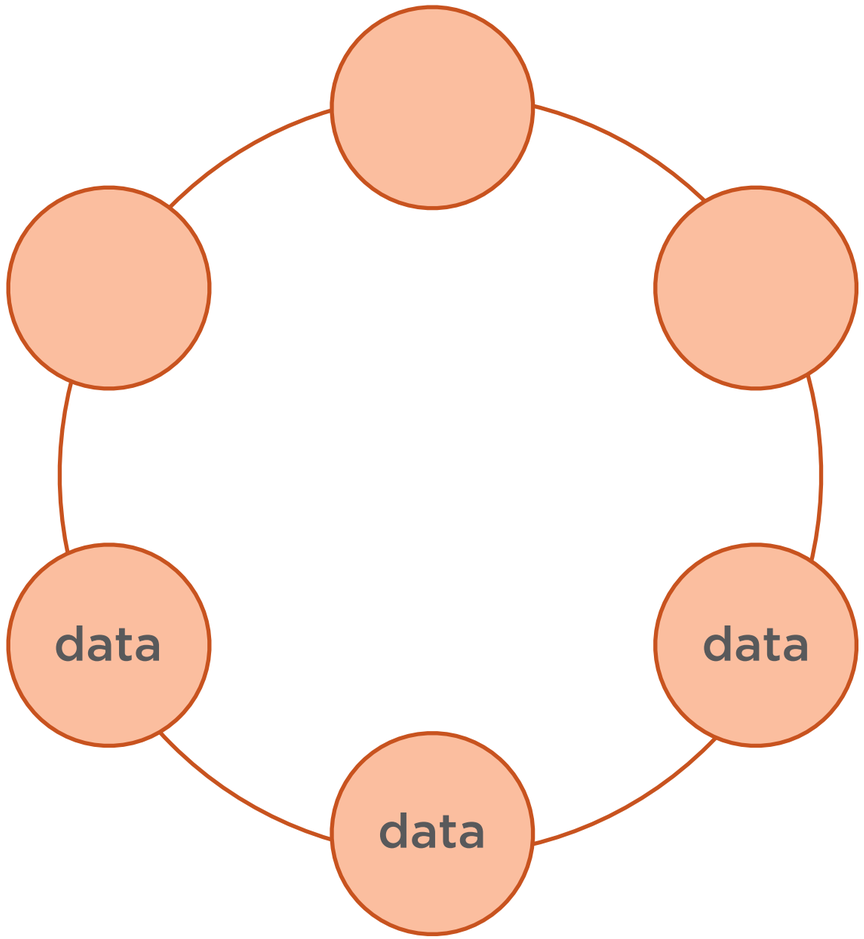
Tombstones



Tombstones


delete author from…
ack
ack
tomb
data
stone
data
tomb
data
stone

Tombstones



gc_grace_seconds

(default is 10 days)

Demo

Populate our courses table

Examine the “writetime” function

Leverage TTLs in a users table

Counters

Creating a table that includes a counter

```
CREATE TABLE pluralsight.ratings (
  course_id varchar PRIMARY KEY,
  ratings_count counter,
  ratings_total counter
);
```

Incrementing a counter

```
UPDATE pluralsight.ratings
SET ratings_count = ratings_count + 1,
    ratings_total = ratings_total + 4
WHERE course_id = ‘cassandra-developers’;
```

Aggregate Functions

Includes: COUNT, MIN, MAX, SUM, and AVG

```
CREATE TABLE pluralsight.ratings (
  course_id varchar,
  user_id varchar,
  rating float,
  PRIMARY KEY (course_id, user_id)
);

SELECT min(rating), max(rating), count(rating), avg(rating)
FROM ratings WHERE course_id = ‘cassandra-developers’;
```

Demo

Create a ratings table using counters

Use aggregate functions with ratings

Conclusion

CQL for interacting with Cassandra

Creating keyspaces and tables

Basic data types

Selects, inserts, updates and deletes

TTLs and tombstones

Counters

Aggregate functions