

# Multi-row Partitions

---



**Paul O'Fallon**

@paulofallon

# Summary

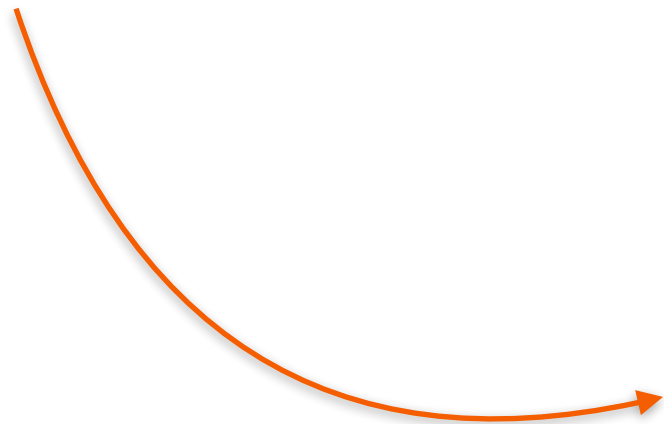
**Composite keys**

**Static columns**

**Time series data**

So Far...

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



```
CREATE TABLE courses (  
    id varchar,  
    PRIMARY KEY (id)  
);
```

# Keys, Keys and More Keys!

**PRIMARY KEY** (`id`)

**PRIMARY KEY** (`partition_key`)

**PRIMARY KEY** (`partition_key, clustering_key`)  
**composite key**

**ONE ROW  
PER PARTITION**



# Composite Primary Keys

**PRIMARY KEY** (*p\_key*, *c\_key*)

**PRIMARY KEY** (*p\_key*, *c\_key*<sub>1</sub>, ... *c\_key*<sub>N</sub>)

# Composite Primary Keys

**PRIMARY KEY** ( $p\_key$ ,  $c\_key$ )

**PRIMARY KEY** ( $p\_key$ ,  $c\_key_1, \dots c\_key_N$ )

**PRIMARY KEY** ( $(p\_key_1, \dots p\_key_N)$ ,  $c\_key_1, \dots c\_key_N$ )

# Composite Primary Keys





























**PRIMARY KEY** ( $p\_key$ ,  $c\_key$ )

**PRIMARY KEY** ( $p\_key$ ,  $c\_key_1$ , ...  $c\_key_N$ )

**PRIMARY KEY** ( $(p\_key_1, \dots p\_key_N)$ ,  $c\_key_1, \dots c\_key_N$ )

**PRIMARY KEY** ( $(p\_key_1, \dots p\_key_N)$ )

# Courses, Now with Modules!

Table of contents	Description	Exercise files	Related Courses
This course is part of:  Python Path <span>Expand All</span>			
	Course Overview		2m 10s 
	Advanced Flow Control		42m 59s 
	Byte-oriented Programming		42m 6s 
	Object Internals and Custom Attributes		30m 51s 
	Descriptors		22m 9s 
	Instance Creation		10m 47s 
	Metaclasses		36m 22s 
	Class Decorators		11m 57s 
	Abstract Base Classes		33m 56s 



# Revisiting Our Courses Schema

```
CREATE TABLE courses (  
    id varchar,  
    name varchar,  
    PRIMARY KEY (id)  
);
```

```
CREATE TABLE courses (  
    id varchar,  
    name varchar,  
    module_id int,  
    module_name varchar,  
    PRIMARY KEY (id, module_id)  
);
```

# Revisiting Our Courses Schema

## Inserting

```
INSERT INTO courses (id, name, module_id, module_name)
VALUES (
    'advanced-python', 'Advanced Python', 1,
    'Course Overview');
```

```
INSERT INTO courses (id, name, module_id, module_name)
VALUES (
    'advanced-python', 'Advanced Python', 2,
    'Advanced Flow Control');
```

# Revisiting Our Courses Schema

## Selecting

**SELECT \* FROM courses WHERE id = 'advanced-python';** ← two rows

**SELECT \* FROM courses  
WHERE id = 'advanced-python' AND module\_id = 1;** ← one row

**SELECT \* FROM courses WHERE id = 'advanced-python'  
ORDER BY module\_id DESC;** ← two rows

# Ordering By Clustering Key

id='advanced-python'

module\_id=1

name	Advanced Python
module_name	Course Overview

module\_id=2

name	Advanced Python
module_name	Advanced Flow Control

# Demo

**Add modules to our courses schema**

**Leverage a clustering key**

# Static Columns

```
CREATE TABLE courses (  
    id varchar,  
    name varchar STATIC,  
    module_id int,  
    module_name varchar,  
    PRIMARY KEY (id, module_id)  
);
```

# Without Static Columns

id='advanced-python'

module_id=1	
name	Advanced Python
module_name	Course Overview

module_id=2	
name	Advanced Python
module_name	Advanced Flow Control

# Static Columns

id='advanced-python'	name		Advanced Python
	module_id=1		
	module_name		Course Overview
	module_id=2		
module_name		Advanced Flow Control	



# Adding Data with Static Columns

```
INSERT INTO courses (id, name)  
VALUES ( 'advanced-python', 'Advanced Python');
```

```
INSERT INTO courses (id, module_id, module_name)  
VALUES ( 'advanced-python', 1, 'Course Overview');
```

```
UPDATE courses  
SET module_name='Advanced Flow Control'  
WHERE id='advanced-python' AND module_id=2;
```

**SELECT  
REMAINS  
THE SAME**

# Demo

**Convert course fields to static columns**

**Select course and module-level data**

# Time Series Data



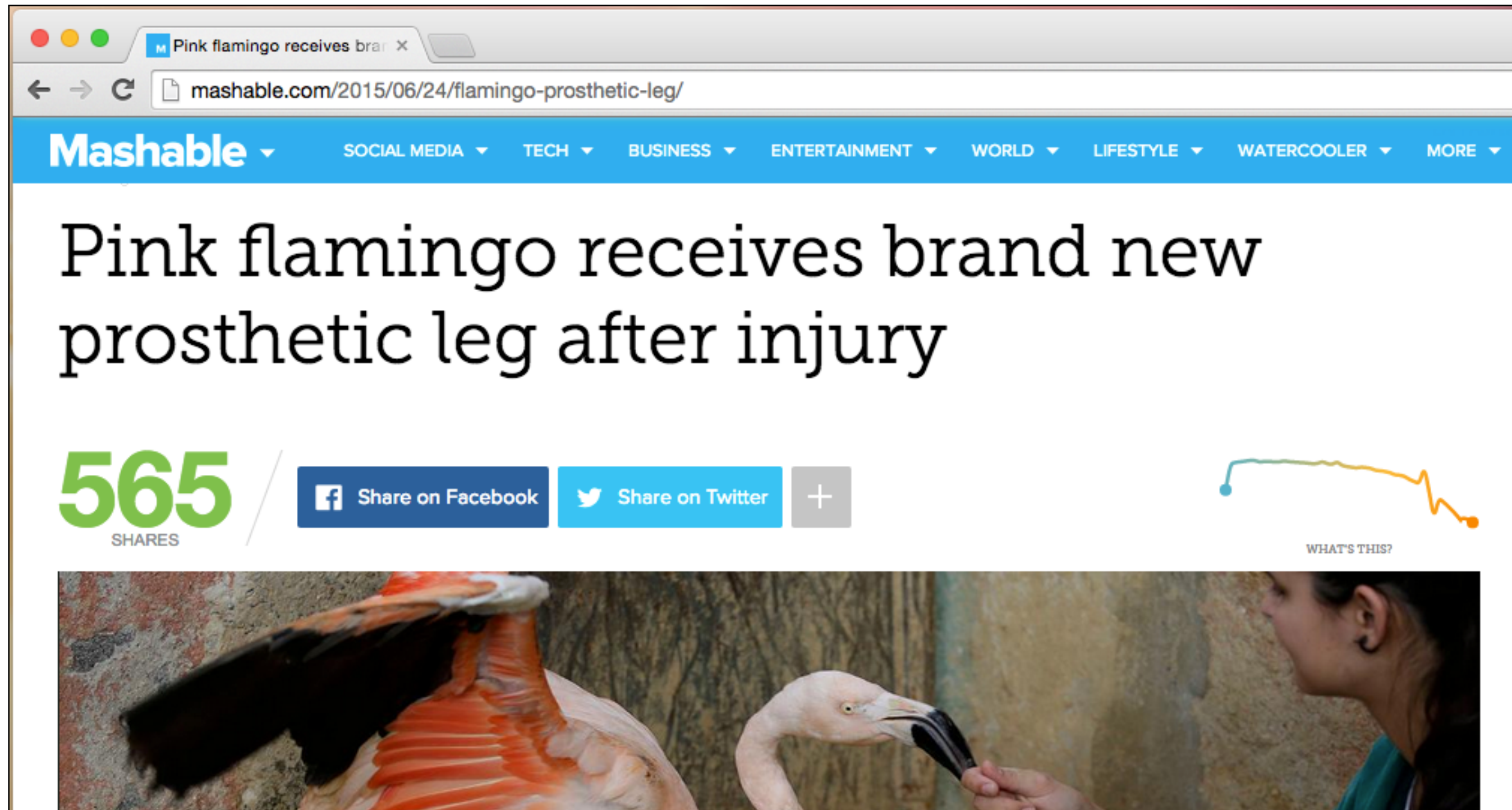
# TimeUUID Data Type

45b94a50-12e5-11e5-9114-091830ac5256

**Version 1 UUID comprised of:**

- The number of 100 ns intervals since UUID epoch
- MAC address
- Clock sequence number to prevent duplicates

# TimeUUID Use Case



The screenshot shows a web browser window with a single tab titled "Pink flamingo receives brand new prosthetic leg". The address bar displays the URL "mashable.com/2015/06/24/flamingo-prosthetic-leg/". The Mashable website header is visible, featuring the logo and navigation links for Social Media, Tech, Business, Entertainment, World, Lifestyle, Watercooler, and More. The main headline reads "Pink flamingo receives brand new prosthetic leg after injury". Below the headline, a green "565" is displayed with "SHARES" underneath. To the right of the share count are buttons for "Share on Facebook", "Share on Twitter", and a plus sign for additional sharing options. A small, colorful line graph with the text "WHAT'S THIS?" is positioned to the right of the share buttons. The article's main image shows a pink flamingo with a prosthetic leg being held by a person's hand.

Pink flamingo receives brand new prosthetic leg after injury

565  
SHARES

Share on Facebook Share on Twitter

WHAT'S THIS?

# TimeUUID Use Case

```
CREATE TABLE course_page_views (  
    course_id text,  
    view_id timeuuid,  
    PRIMARY KEY (course_id, view_id)  
) WITH CLUSTERING ORDER BY (view_id DESC);
```

# TimeUUID Functions

**now**

```
INSERT INTO course_page_views (course_id, view_id)  
VALUES ( 'advanced-python', now());
```

**dateOf / unixTimestampOf**

```
SELECT course_id, dateOf(view_id)  
FROM course_page_views WHERE course_id = 'advanced-python';
```

# TimeUUID Functions

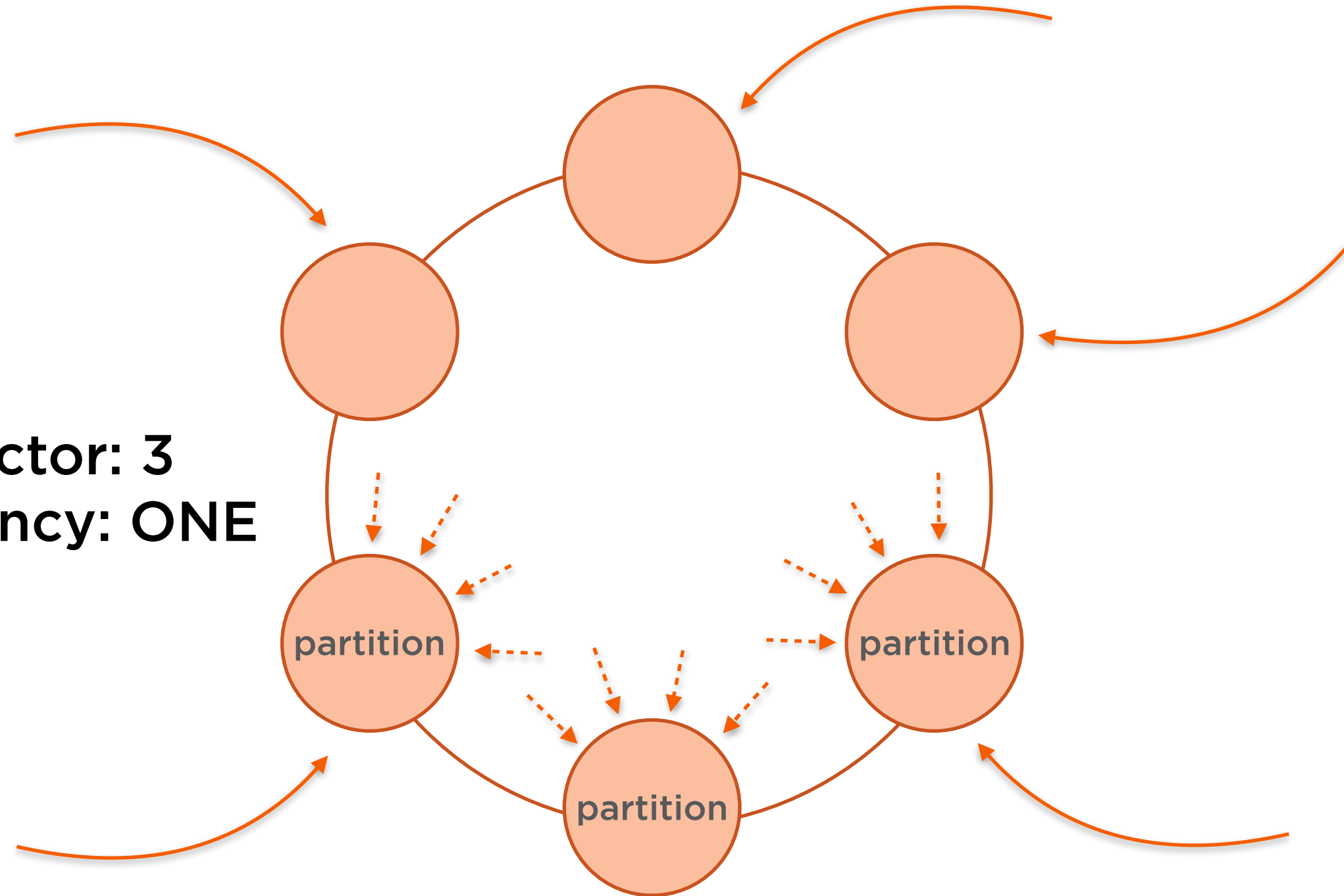
## minTimeuuid / maxTimeuuid

```
SELECT dateOf(view_id)  
FROM course_page_views  
WHERE course_id = 'advanced-python'  
AND view_id >= maxTimeuuid('2019-11-01 00:00+0000')  
AND view_id < minTimeuuid('2019-12-01 00:00+0000')
```



# Storing Data with a Clustering Key

**replication\_factor: 3**  
**write consistency: ONE**



# Storing Data with a Clustering Key

course\_id='advanced-python'

view\_id='2d0e7b64-1bbf-11ea-978f-2e728ce88125'

# Storing Data with a Clustering Key

course\_id='advanced-python'

view\_id='62801640-1bbf-11ea-978f-2e728ce88125'

view\_id='2d0e7b64-1bbf-11ea-978f-2e728ce88125'

# Storing Data with a Clustering Key

course\_id='advanced-python'

view\_id='62801640-1bbf-11ea-978f-2e728ce88125'

view\_id='74070a36-1bbf-11ea-978f-2e728ce88125'

view\_id='2d0e7b64-1bbf-11ea-978f-2e728ce88125'

# Demo

**Create a table for course page views**

**Insert data with a TimeUUID and TTL**

**Track the last time a page was viewed**

# Bucketing Time Series Data

partition-key

**A maximum of 2 billion cells**  
(rows x columns)

**Must fit on a single node**

# Bucketing Time Series Data



# Bucketing Use Case

```
CREATE TABLE course_page_views (  
    bucket_id text,  
    course_id text,  
    view_id timeuuid,  
    PRIMARY KEY ((bucket_id, course_id), view_id)  
) WITH CLUSTERING ORDER BY (view_id DESC);
```



# Inserting Bucketed Data

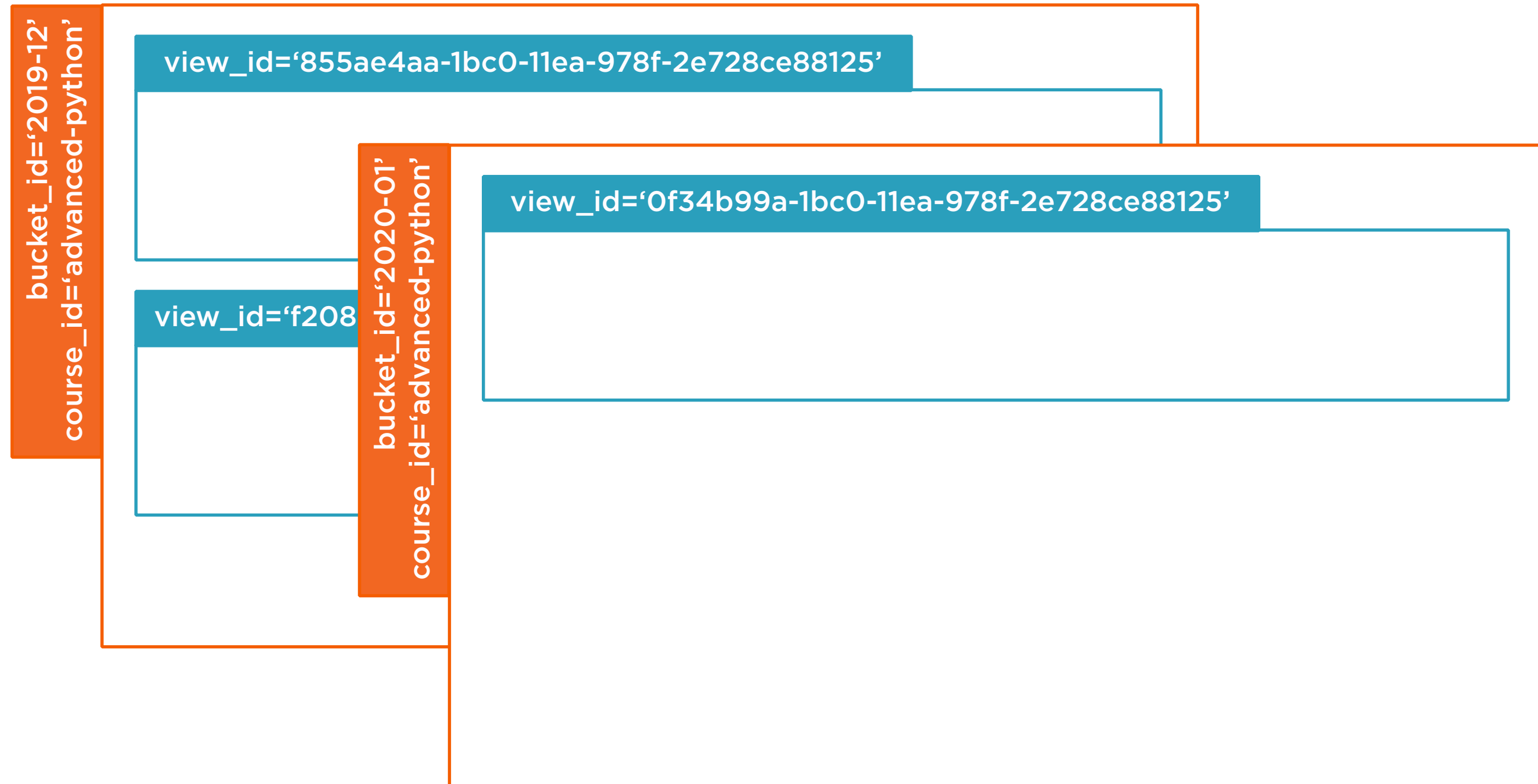
```
INSERT INTO course_page_views  
    (bucket_id, course_id, view_id)  
VALUES  
    ( '2019-12', 'advanced-python', now() );
```

# Inserting Bucketed Data

bucket\_id='2019-12'  
course\_id='advanced-python'

view\_id='f208cab0-13bc-11e5-b559-4b636433f200'

# Inserting Bucketed Data



# Inserting Bucketed Data



# Demo

**Use a bucket id to segment page views**

# Conclusion

**Clustering and composite primary keys**

**Static columns**

**Time series data**

**TimeUUID data type**

**TTL vs. bucketing**