The Cassandra Data Model

Lets take a bottom-up approach to understand the Cassandra data model

COLUMN

Basic data structure of Cassandra

We need the following parameters for creating a column

name

data type

Simple data types

int
float
double
boolean
text

COLUMN

We need the following parameters for creating a column

name

data type

int
float
double
boolean
text

Advanced data types

blob
counter
uuid
timestamp

COLUMN

We need the following parameters for creating a column

name

data type

int
float
double
boolean
text

blob
counter
uuid
timestamp

collection data types

set
list
map

COLUMN

Basic data structure of Cassandra

name

data type

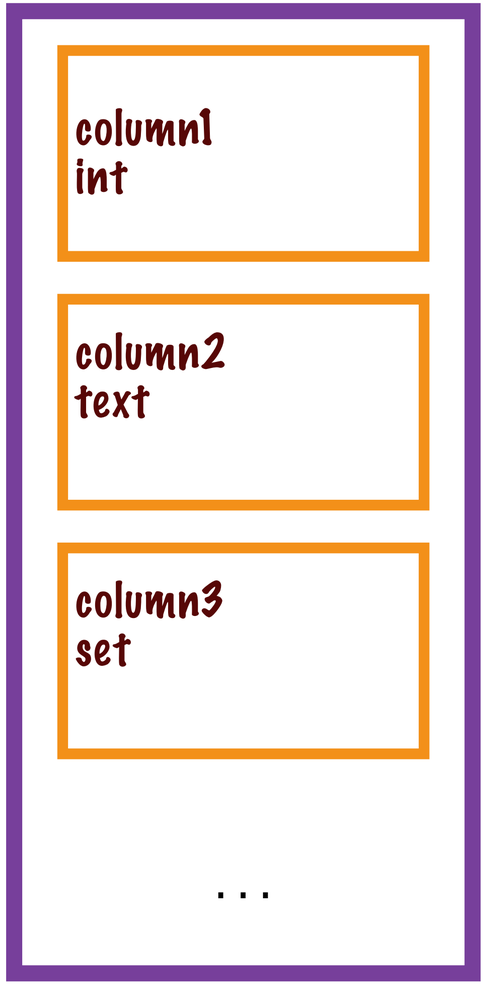
Data for a column is stored as a column - value pair in the database

column key1
column value1

data representation of a column

COLUMN FAMILY

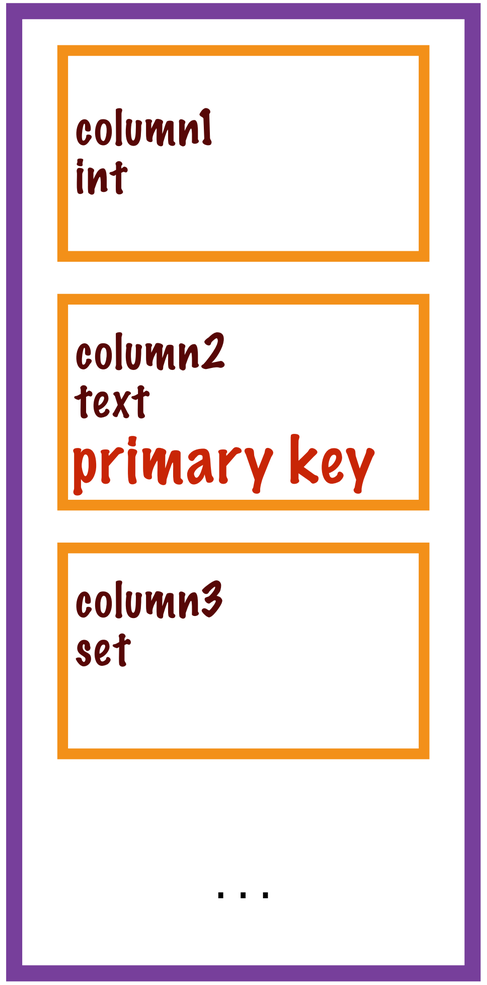
Collection of one or more columns



Columns can be added to the family at any point of time

COLUMN FAMILY

Collection of one or more columns

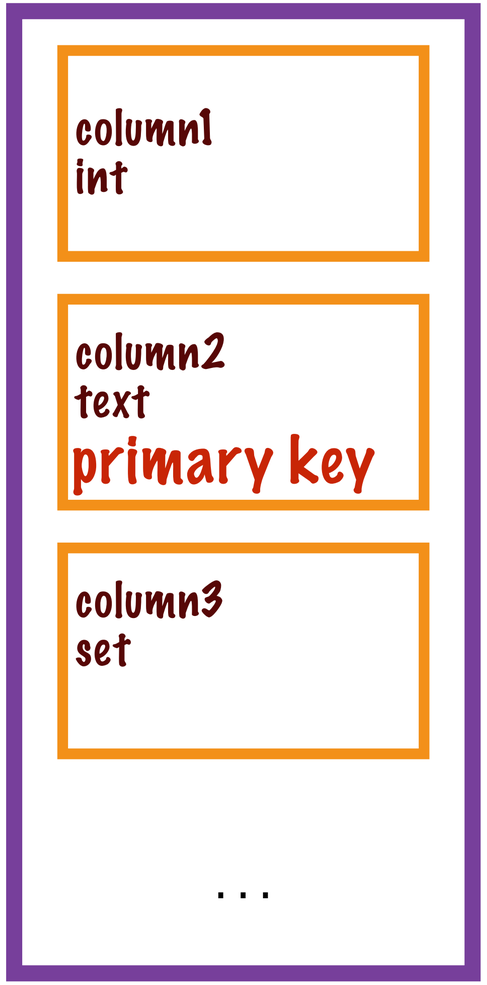


Columns can be added to the family at any point of time

Every columnfamily has a **primary key** to uniquely identify rows in it

COLUMN FAMILY

Collection of one or more columns



Columns can be added to the family at any point of time

Every columnfamily has a primary key to uniquely identify rows in it

primary key can be composed of multiple columns

composite key

COLUMN FAMILY

Data representation of columnfamily

Collection of column value pairs
form a row

Earlier we saw that a
column value pair is stored as

column key1
column value1

COLUMN FAMILY

Data representation of columnfamily

value of the primary key column is the row key

Collection of column value pairs form a row

row key value1	column key1	column key2	column key3
	column value1	column value2	column value3

In case of composite keys, the value of the 1st column in the composite key becomes the row key

COLUMN FAMILY

Data representation of columnfamily

a row

row key

row key value1	column key1	column key2	column key3
	column value1	column value2	column value3

Row key + Column Family = Row

COLUMN FAMILY

ATTRIBUTES

keys_cached

Cassandra has a built in key cache and row cache

key cache holds the location of
keys in memory

keys_cached is the number of
keys for which the key cache
will hold the location data

by default the value of
keys_cached is set to 200,000

COLUMN FAMILY

ATTRIBUTES

keys_cached

rows_cached

row cache holds the entire data of the row in memory

best used if you have a small subset of data that is frequently used

rows_cached is the number of rows to be stored in the row cache

COLUMN FAMILY

ATTRIBUTES

keys_cached

rows_cached

preload_row_cache

flag to specify whether you want to pre-populate the row cache

COLUMN FAMILY

ATTRIBUTES

keys_cached

rows_cached

preload_row_cache

SUPER COLUMN FAMILY

A special type of column family

The value is a collection of columns

We can group the columns which we are likely to query together under the same SCF

A logical grouping of columns which belong together

This optimizes READ operations on the table!

SUPER COLUMN FAMILY

Data representation of a row with SCF

row key value1	super column key1		super column key3	
	column key1	column key2	column key3	column key4
	column value1	column value2	column value3	column value4

Super Column Family is a subset of the column family in a row

This row has 2 SCFs

SUPER COLUMN FAMILY

Data representation of a row with SCF

key

row key value1	super column key1		super column key3	
	column key1	column key2	column key3	column key4
	column value1	column value2	column value3	column value4

value

The key in a super column family (SCF) is name of the SCF

SUPER COLUMN FAMILY

super column key1			
column key1	column key2	column key3	....
column value1	column value2	column value3	....

Theoretically there is **no limit** to the number of columns that can belong to an SCF

SUPER COLUMN FAMILY

~~get("super column key1",
"column key 2")~~

BUT..

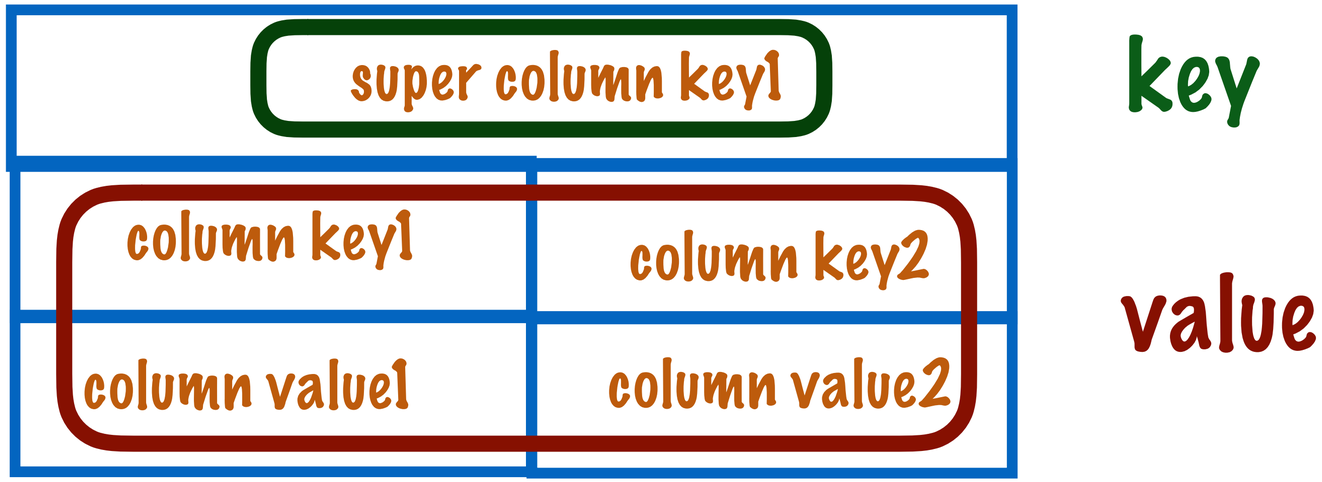
super column key1			
column key1	column key2	column key3	....
column value1	column value2	column value3	....

Cassandra doesn't index the columns in SCF

We cannot query individual columns in SCF

SUPER COLUMN FAMILY

BUT..



When we query for "super column key1", all the columns associated with the SCF are loaded in memory

SUPER COLUMN FAMILY

BUT..

super column key1	
column key1	column key2
column value1	column value2

At runtime, qps for read operation on SCF will be huge - in the tens of thousands

SUPER COLUMN FAMILY

BUT..

super column key1	
column key1	column key2
column value1	column value2

With SCF loading all columns in memory, application will run out of memory

SUPER COLUMN FAMILY

BUT..

super column key1	
column key1	column key2
column value1	column value2

So avoid designing a data model with a Super Column Family

SUPER COLUMN FAMILY

BUT..

super column key1	
column key1	column key2
column value1	column value2

Instead a composite key should be used when we want to group the columns

KEYSPACE

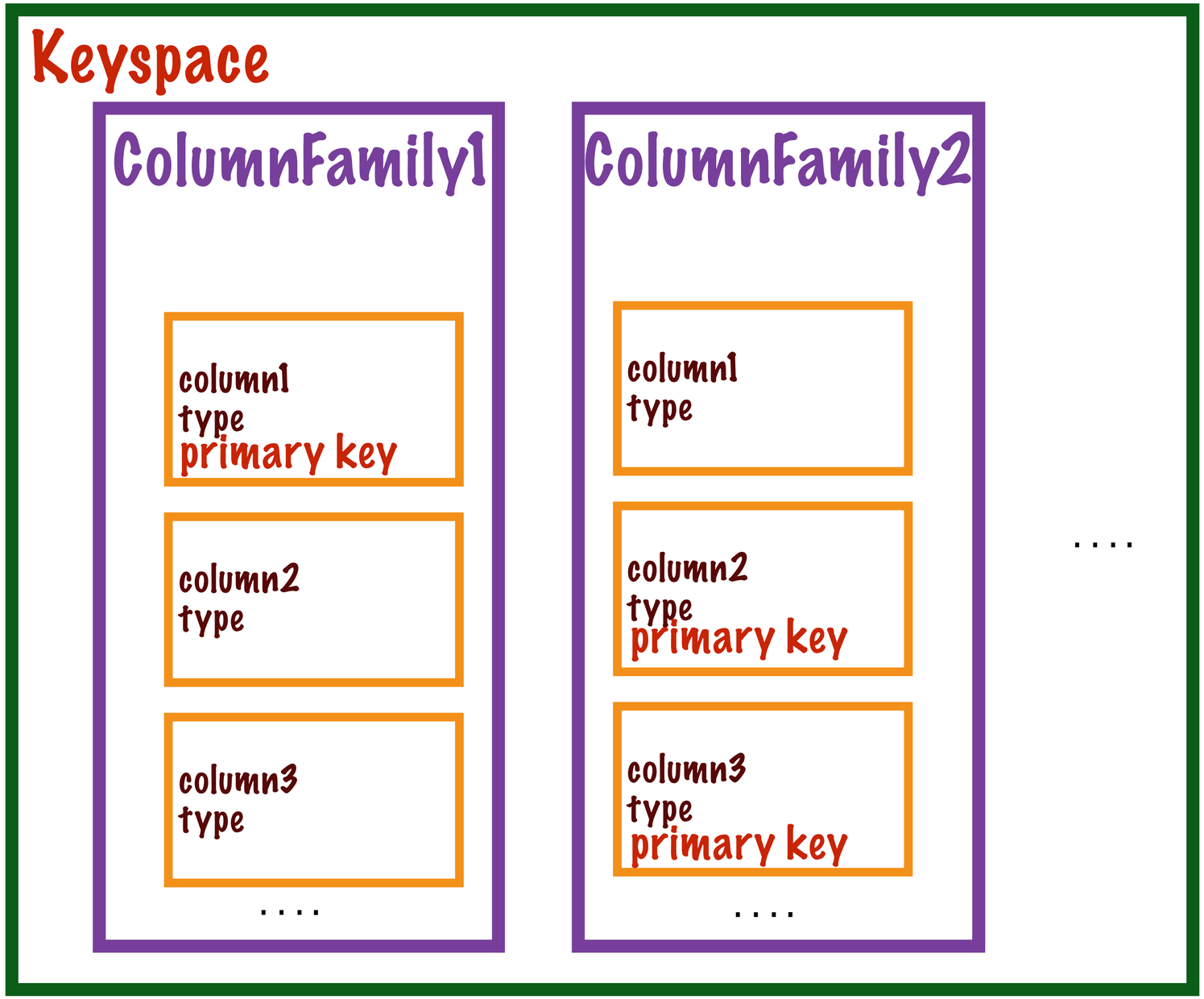
Collection of one or more column families

Keyspace is the outermost
container of data

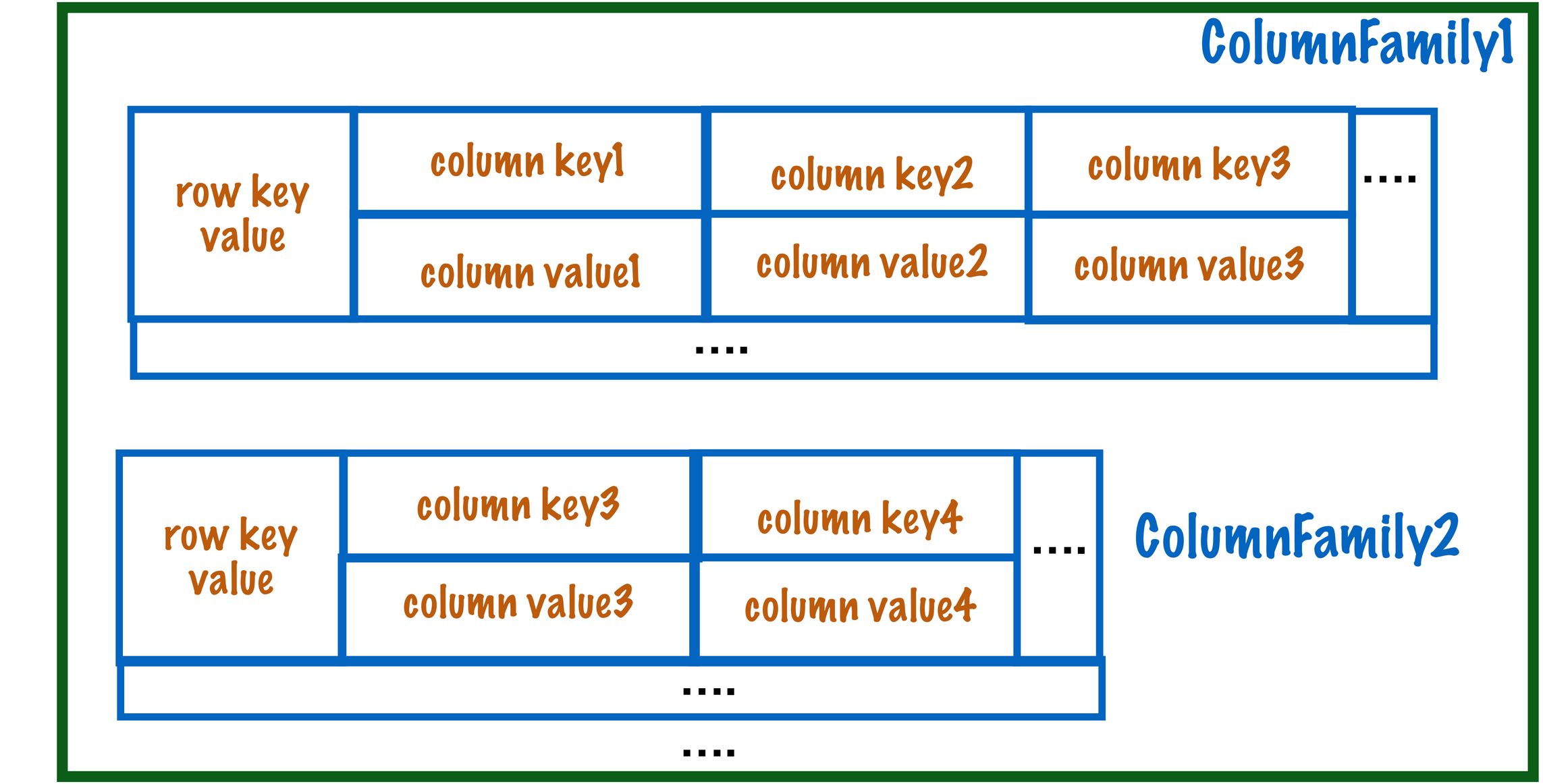
We can have more than 1
keyspace in a cluster

Lets see the full picture..

Cassandra


Keyspace
ColumnFamily1
column1
type
primary key
column2
type
column3
type
....
ColumnFamily2
column1
type
column2
type
primary key
column3
type
primary key
....
....

Data representation of a keyspace



A quick analogy between a Relational DB and Cassandra

relational		cassandra
database	⟷	keyspace
table	⟷	columnfamily
primary key	⟷	row key
column name	⟷	column name
column value	⟷	column value

A quick analogy between a Relational DB and Cassandra

This is just for better understanding of the scope of each of the constructs

Do not use this when you are designing your database

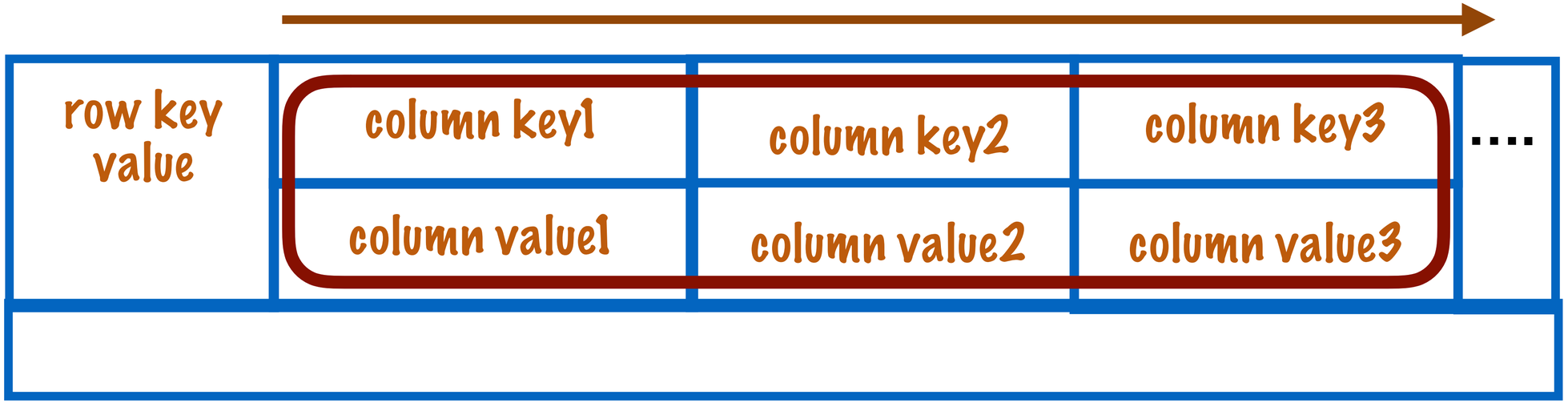
Data In Cassandra

A nested map is a more accurate analogy for understanding how data is stored in cassandra

Data In Cassandra

A nested map is a more accurate analogy for understanding how data is stored in cassandra

columns are Stored in sorted order by column key



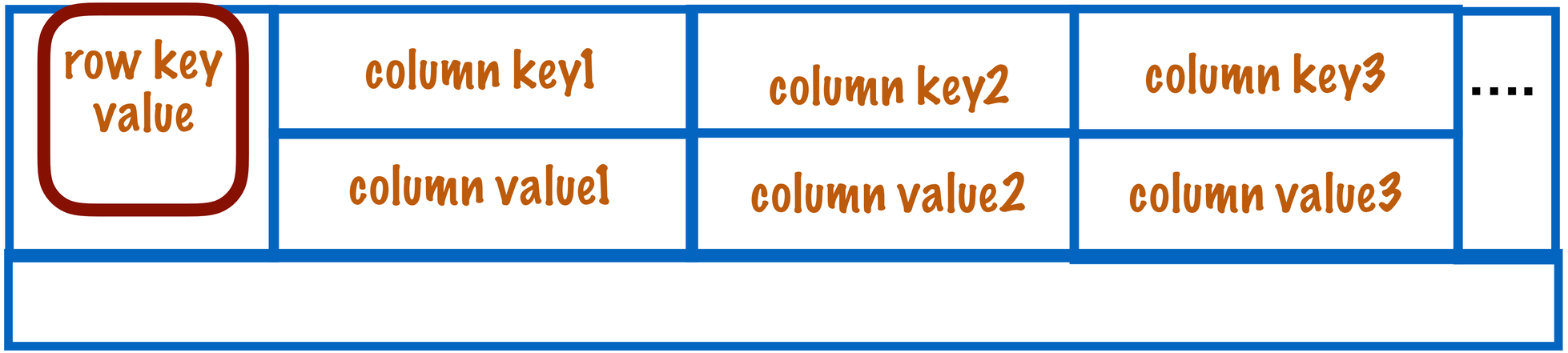
SortedMap<ColKey, ColValue>

Data In Cassandra

A nested map is a more accurate analogy for understanding how data is stored in cassandra

columns are Stored in sorted order by column key

row key points to this map



RowKey, SortedMap<ColKey, ColValue>

Data In Cassandra

A nested map is a more accurate analogy for understanding how data is stored in cassandra

columns are Stored in sorted order by column key

multiple row-> column entries

row key value	column key1	column key2	column key3	....
	column value1	column value2	column value3	

Map<RowKey, SortedMap<Colkey, ColValue>>

Data In Cassandra

A nested map is a more accurate analogy for understanding how data is stored in cassandra

Map<RowKey, SortedMap<Col key, ColValue>>

A Map allows us to look up keys efficiently

Since the columns are stored in a sorted manner, we can perform a range scan on them