Apache Cassandra

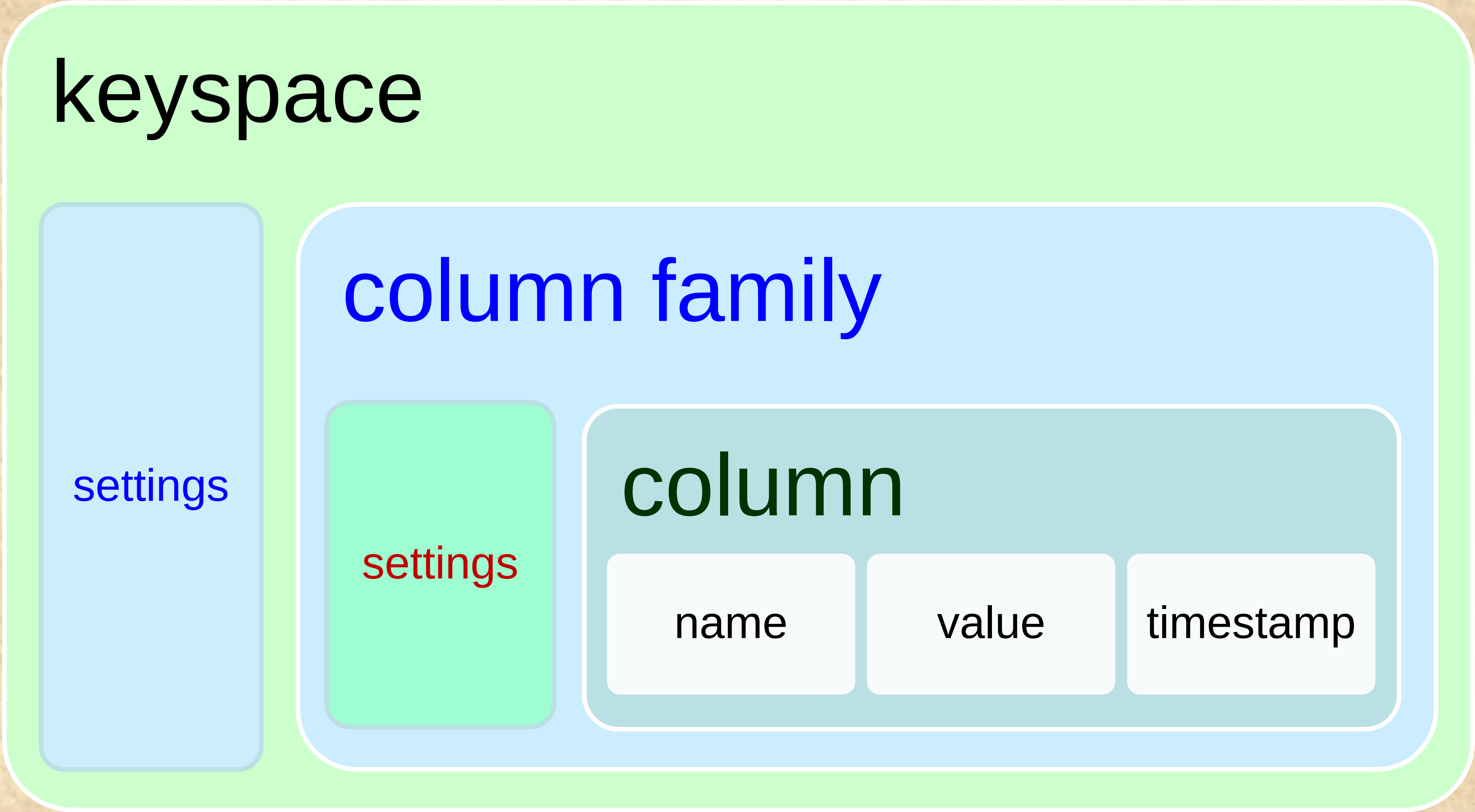
Introduction

- Apache Cassandra is a free, open source, distributed data storage system that differs sharply from relational database management systems.
- Cassandra was created to power the Facebook Inbox Search
- Facebook open-sourced Cassandra in 2008 and became an Apache Incubator project
- In 2010, Cassandra graduated to a top-level project, regular update and releases followed.
- Designed to **handle large amount of data** across multiple servers
- Easy to **implement** and **deploy**
- Mimics traditional relational database systems, but with **triggers** and **lightweight transactions**
- Raw, simple data structures
- *Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Netflix, and more*

Data Model : *Key-Value* Model

- Cassandra is a column oriented NoSQL system
- Table is a multi dimensional map indexed by key (row key).
- Column families: sets of key-value pairs
  - column family as a table and key-value pairs as a row (using relational database analogy)
- A row is a collection of columns labeled with a name, value, timestamp

Key-Value **Model**



keyspace

settings

column family

settings

column

name

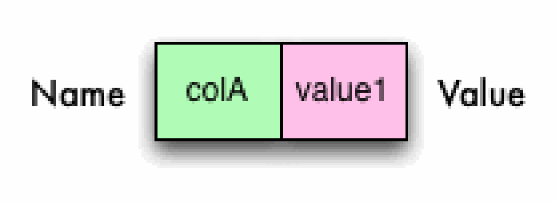
value

timestamp

Certified for IBM DB2 e-business software

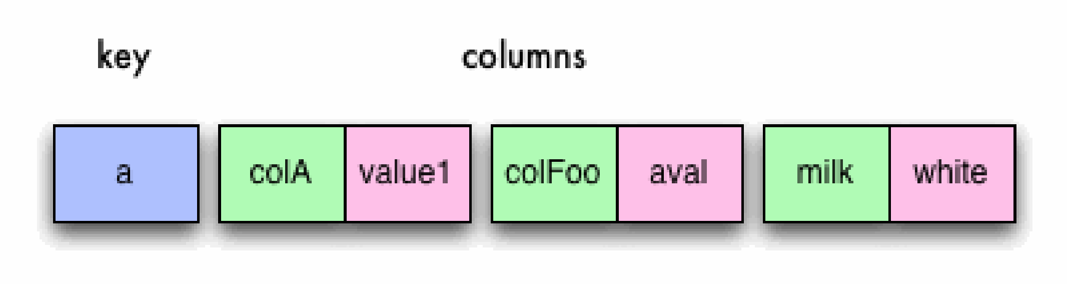
Example

A single column



Name colA value1 Value

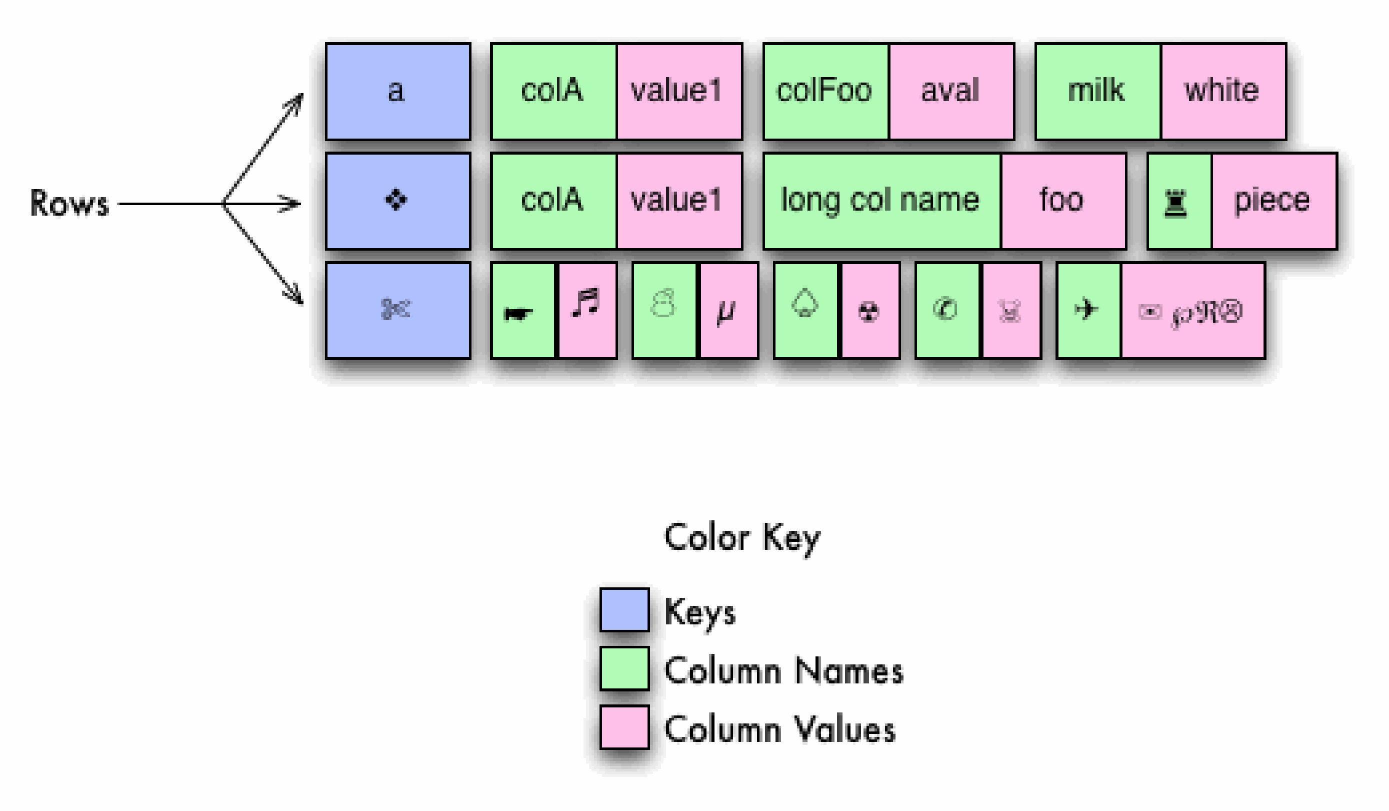
A single row



key columns

a colA value1 colFoo aval milk white

Column family



Rows

a colA value1 colFoo aval milk white

❖ colA value1 long col name foo ♜ piece

✂ ☛ ♬ ☃ μ ♤ ☢ ✆ ♉ ✈ ✉℘ℜ⊗

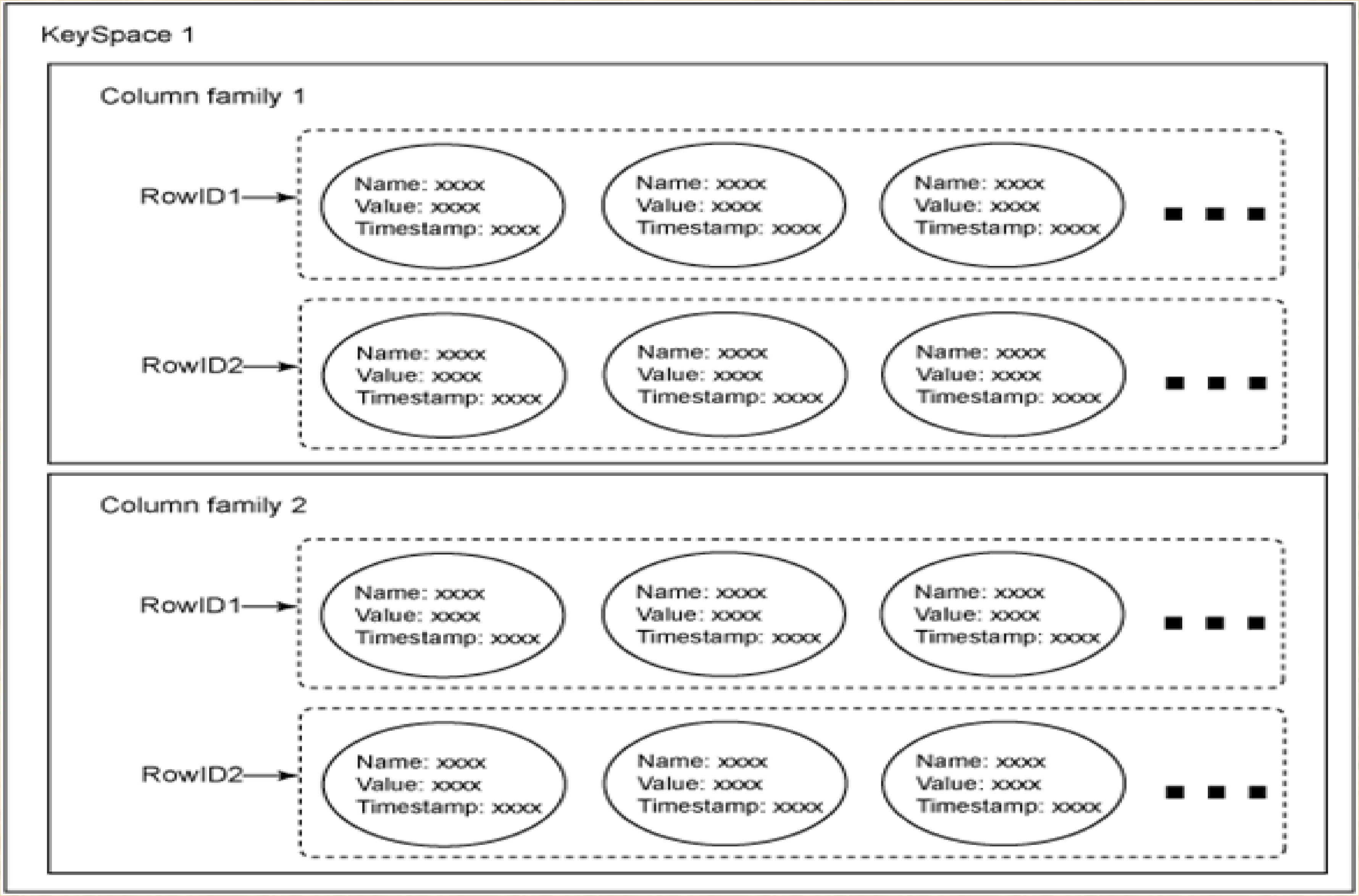
Color Key

🟦 Keys

🟩 Column Names

🟪 Column Values

Example



KeySpace 1

Column family 1

RowID1

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

■ ■ ■

RowID2

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

■ ■ ■

Column family 2

RowID1

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

■ ■ ■

RowID2

Name: xxxx
Value: xxxx
Timestamp: xxxx

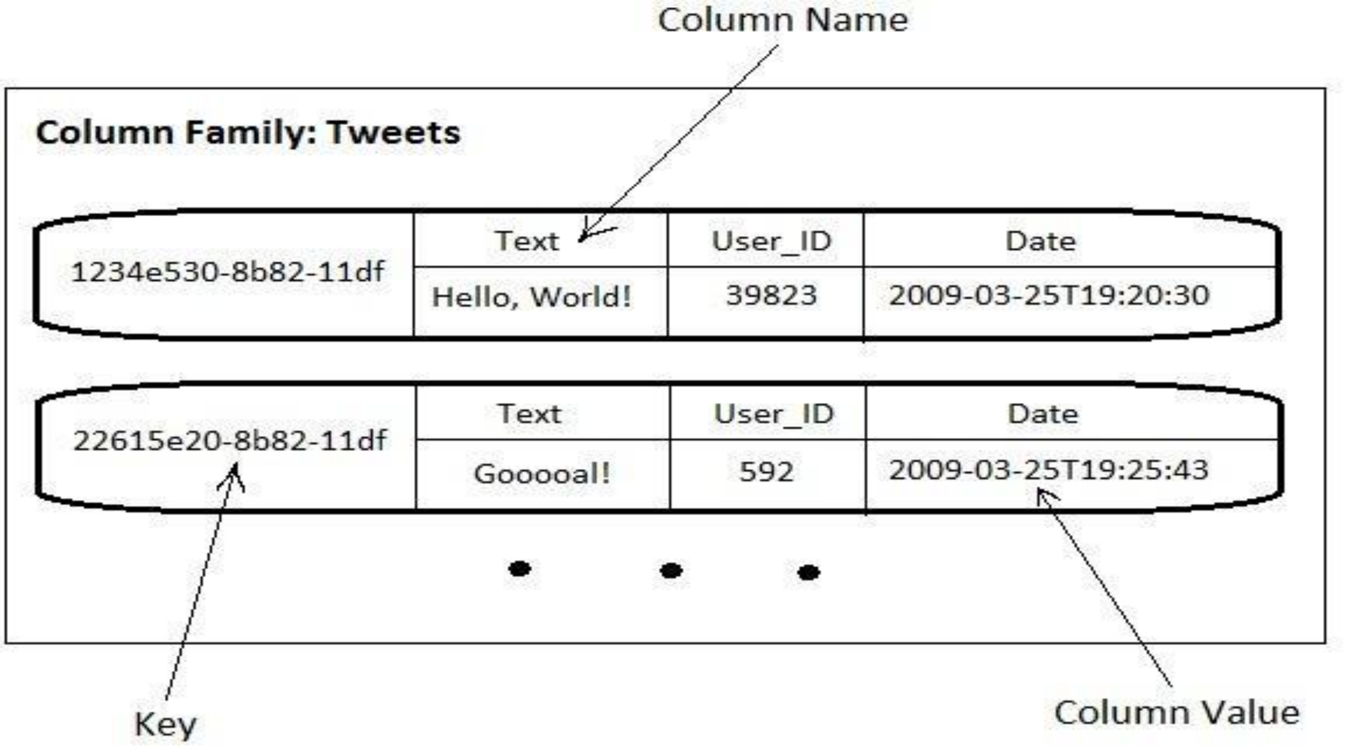
Name: xxxx
Value: xxxx
Timestamp: xxxx

Name: xxxx
Value: xxxx
Timestamp: xxxx

■ ■ ■

Cassandra Row

- the value of a row is itself a sequence of key-value pairs
- such nested key-value pairs are *columns*
- key = column name
- a row must contain at least 1 column



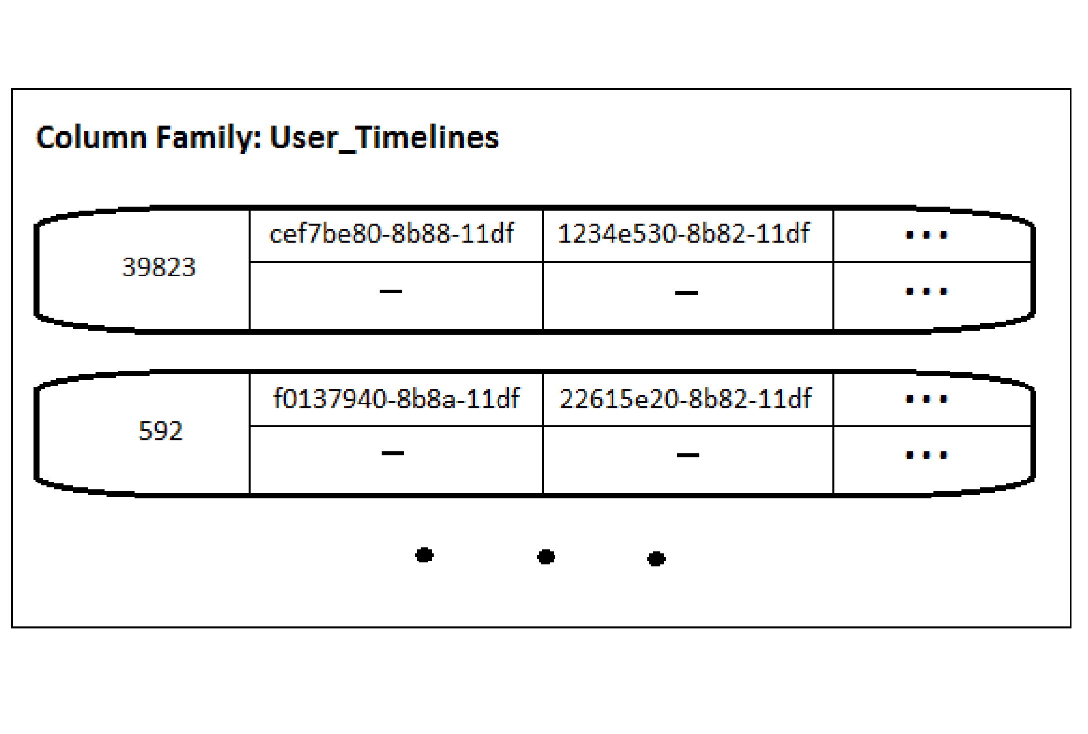
Column Family: Tweets

1234e530-8b82-11df	Text	User_ID	Date
	Hello, World!	39823	2009-03-25T19:20:30

22615e20-8b82-11df	Text	User_ID	Date
	Gooooal!	592	2009-03-25T19:25:43

Column names storing values

- key: User ID
- column names store tweet ID values
- values of all column names are set to “-” (empty byte array) as they are not used



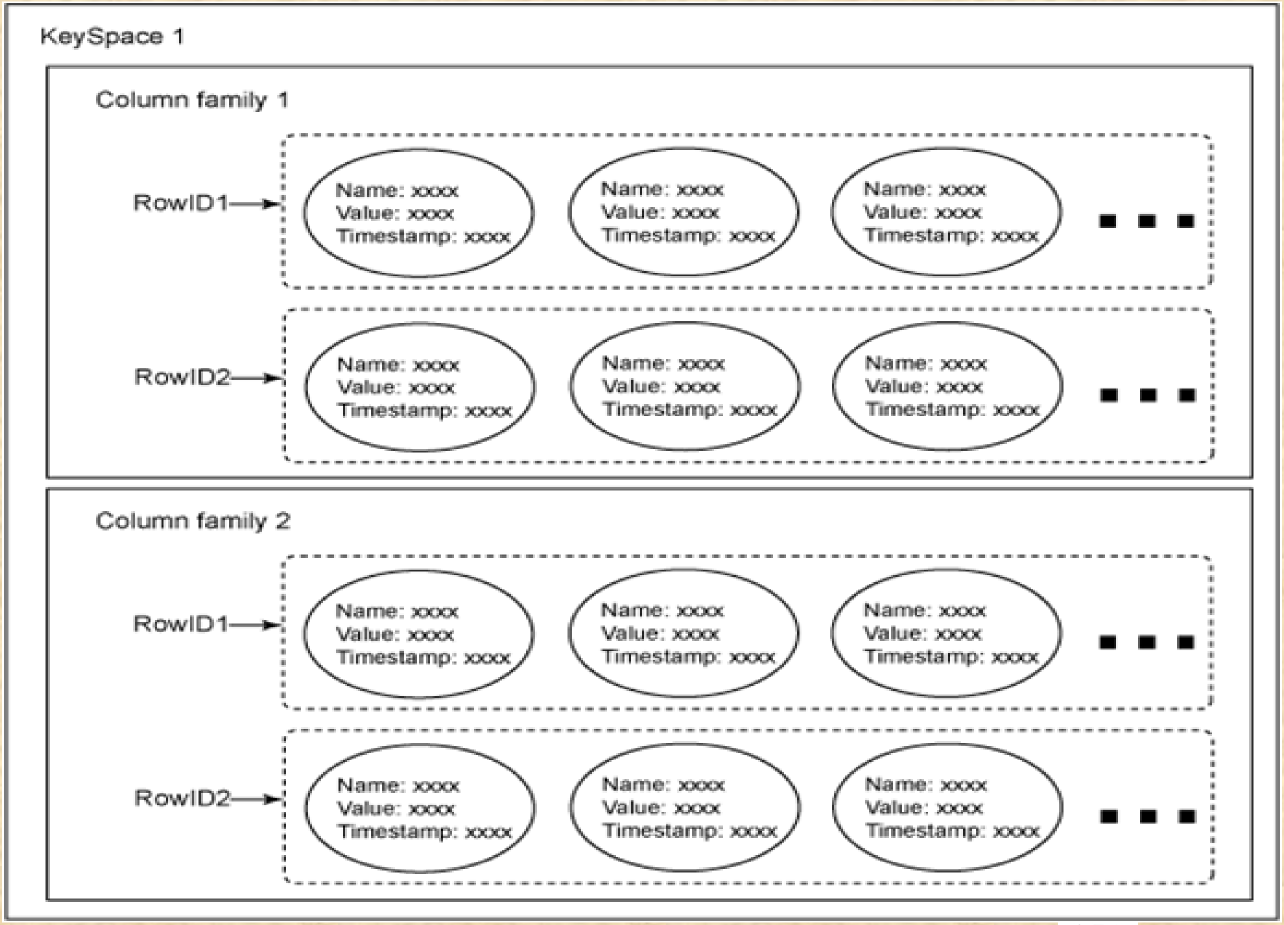
Column Family: User_Timelines

39823	cef7be80-8b88-11df	1234e530-8b82-11df	...
	–	–	...
592	f0137940-8b8a-11df	22615e20-8b82-11df	...
	–	–	...

• • •

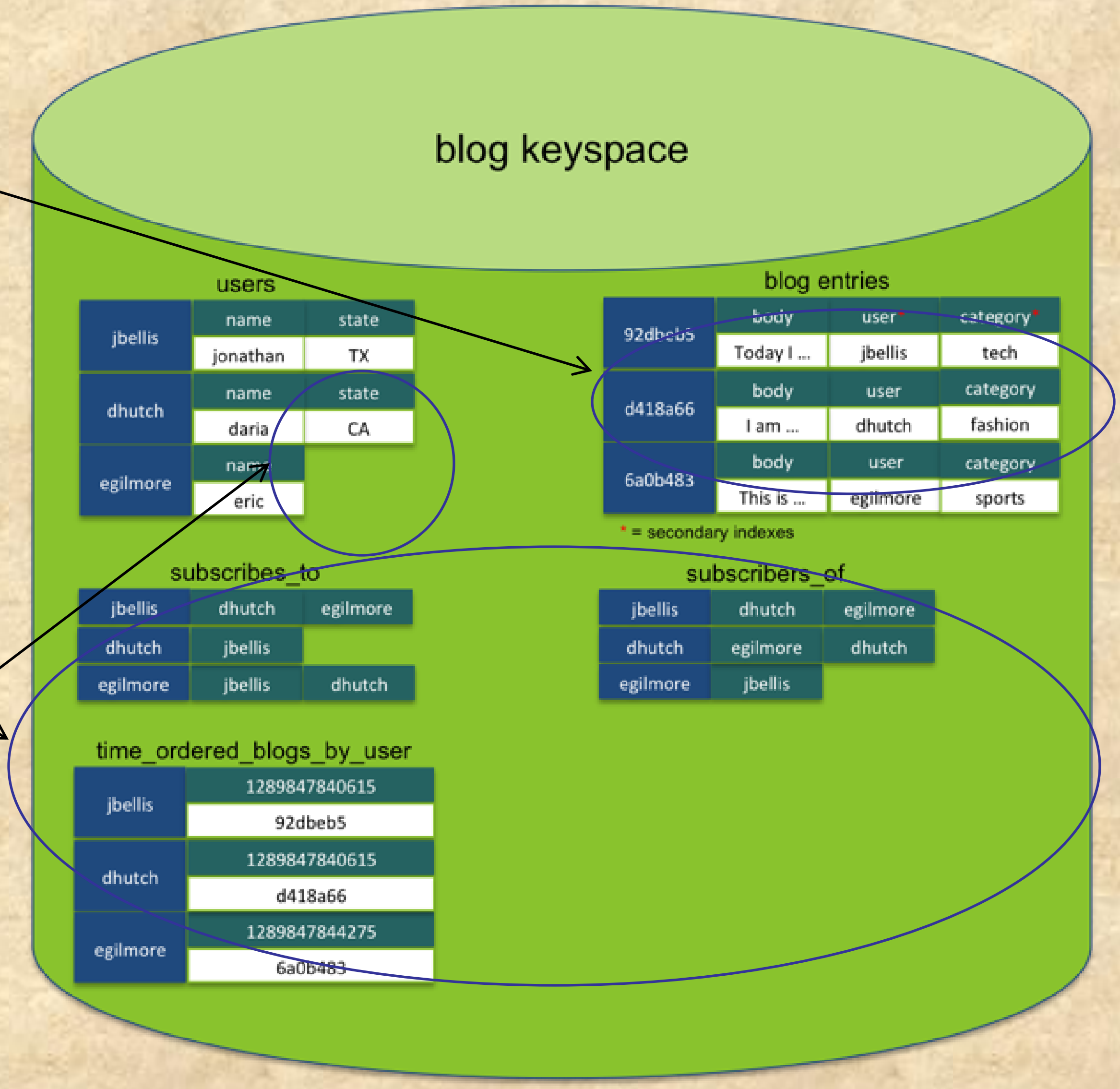
Key Space

- A Key Space is a group of column families together
- It is only a logical grouping of column families and provides an isolated scope for names



Cassandra Data : Storage

- **Column Families:**
  - Like SQL tables
  - but may be unstructured (client-specified)
  - Can have index tables
- **“column-oriented databases”/ “NoSQL”**
  - No schemas
  - Some columns missing from some entries
  - “Not Only SQL”
  - Supports get(key) and put(key, value) operations
  - Often write-heavy workloads



users

jbellis	name jonathan	state TX
dhutch	name daria	state CA
egilmore	name eric	

blog entries

92dbeb5	body Today I ...	user* jbellis	category* tech
d418a66	body I am ...	user dhutch	category fashion
6a0b483	body This is ...	user egilmore	category sports

* = secondary indexes

subscribes_to

jbellis	dhutch	egilmore
dhutch	jbellis	
egilmore	jbellis	dhutch

subscribers_of

jbellis	dhutch	egilmore
dhutch	egilmore	dhutch
egilmore	jbellis	

time_ordered_blogs_by_user

jbellis	1289847840615 92dbeb5
dhutch	1289847840615 d418a66
egilmore	1289847844275 6a0b483

Cassandra Query Language - CQL

- creating a *keyspace* - namespace of tables

```
CREATE KEYSPACE demo
WITH replication = {‘class’: ’SimpleStrategy’,
‘replication_factor’: 3};
```

- to use namespace:

```
USE demo;
```

Cassandra Query Language - CQL

- creating tables

```
CREATE TABLE users(
email varchar,
bio varchar,
birthday timestamp,
active boolean,
time_posted));
PRIMARY KEY (email));
```

```
CREATE TABLE tweets(
email varchar,
time_posted timestamp,
tweet varchar,
PRIMARY KEY (email,
```

- inserting data

```
INSERT INTO users (email, bio, birthday, active)
VALUES (‘john.doe@bti360.com’, ‘BT360 Teammate’, 516513600000, true);
```

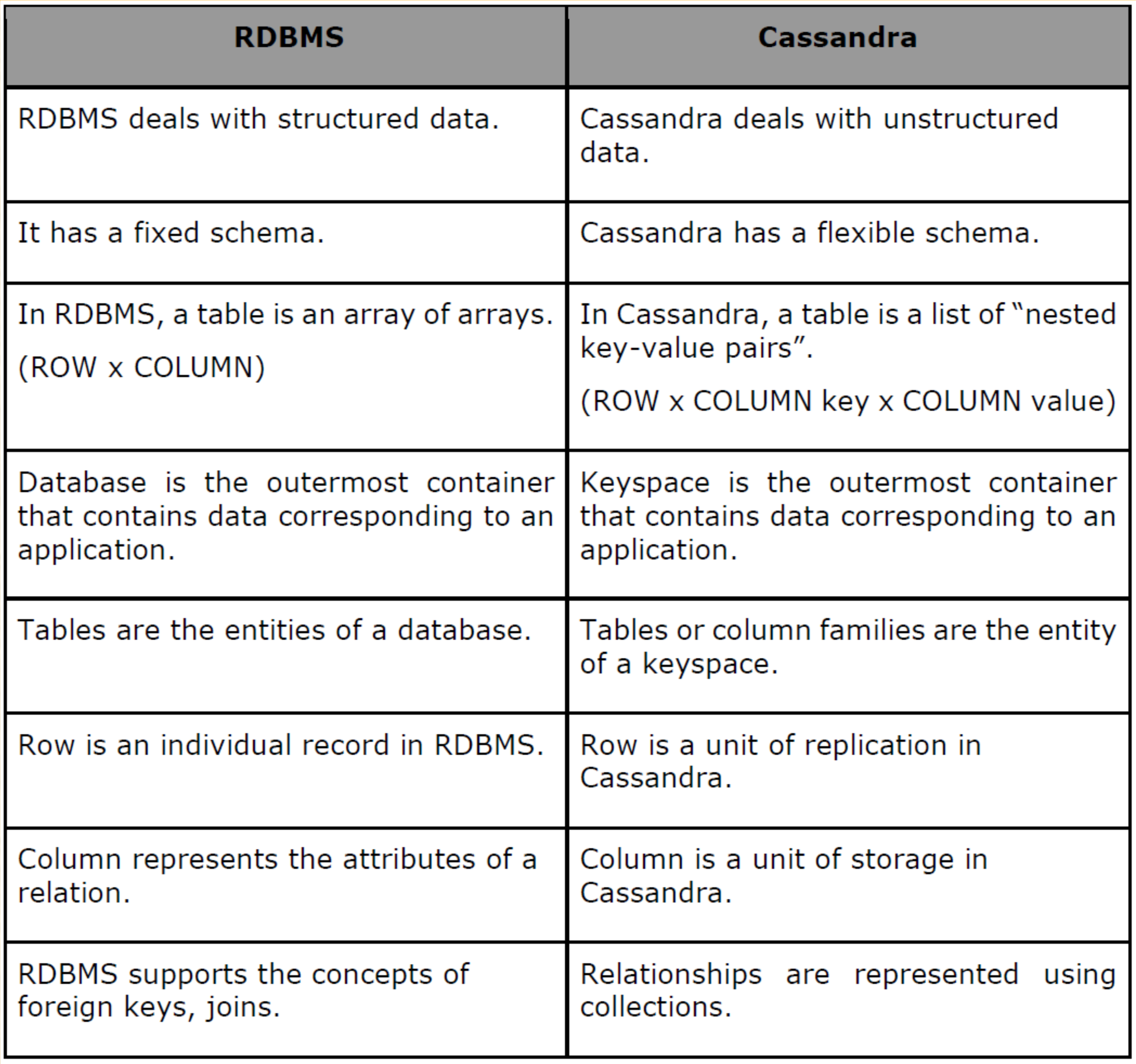
*** timestamp fields are specified in milliseconds since epoch*

- querying tables

SELECT expression reads one or more records from Cassandra column family and returns a result-set of rows

```
SELECT * FROM users;          SELECT email FROM users WHERE active = true;
```

Data Models of Cassandra and RDBMS



RDBMS	Cassandra
RDBMS deals with structured data.	Cassandra deals with unstructured data.
It has a fixed schema.	Cassandra has a flexible schema.
In RDBMS, a table is an array of arrays. (ROW x COLUMN)	In Cassandra, a table is a list of "nested key-value pairs". (ROW x COLUMN key x COLUMN value)
Database is the outermost container that contains data corresponding to an application.	Keyspace is the outermost container that contains data corresponding to an application.
Tables are the entities of a database.	Tables or column families are the entity of a keyspace.
Row is an individual record in RDBMS.	Row is a unit of replication in Cassandra.
Column represents the attributes of a relation.	Column is a unit of storage in Cassandra.
RDBMS supports the concepts of foreign keys, joins.	Relationships are represented using collections.