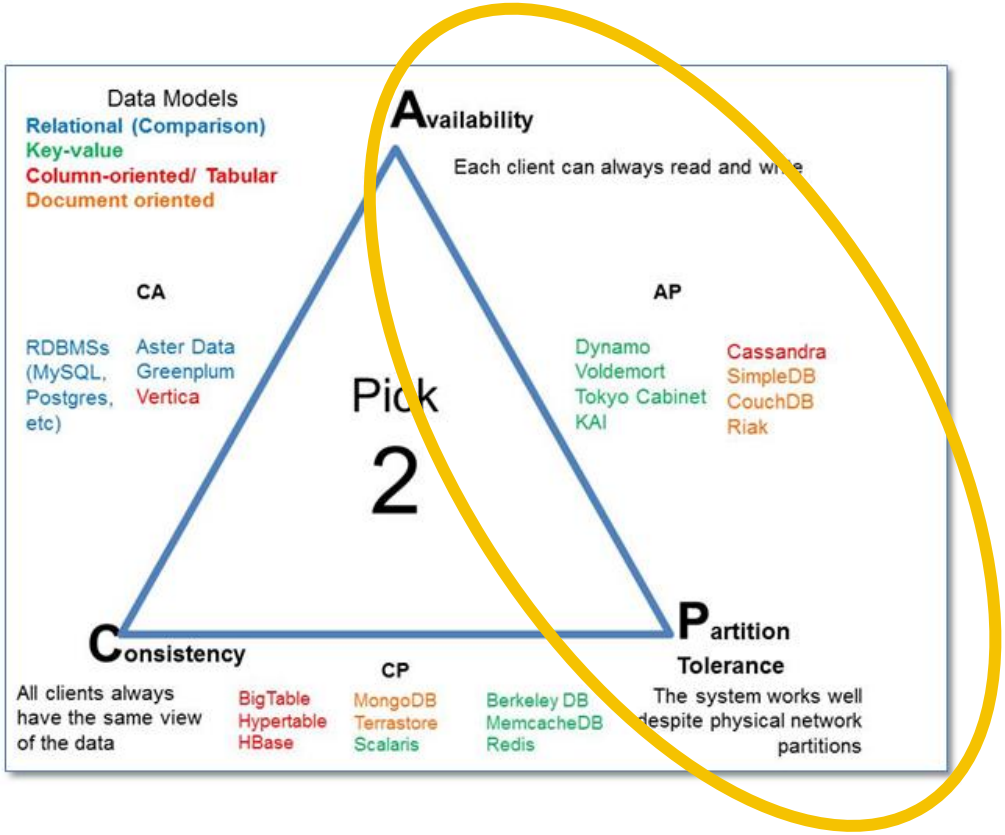
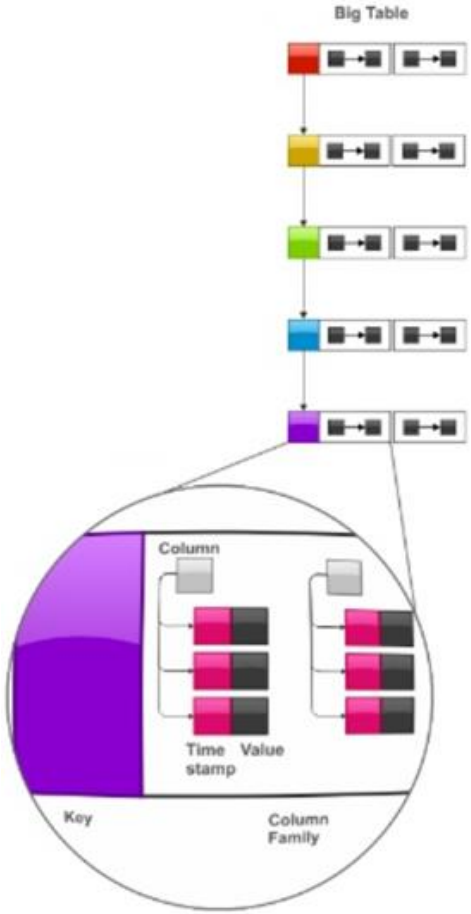
BD2_C1A CASSANDRA MODELADO

SERGIO ÁLVAREZ

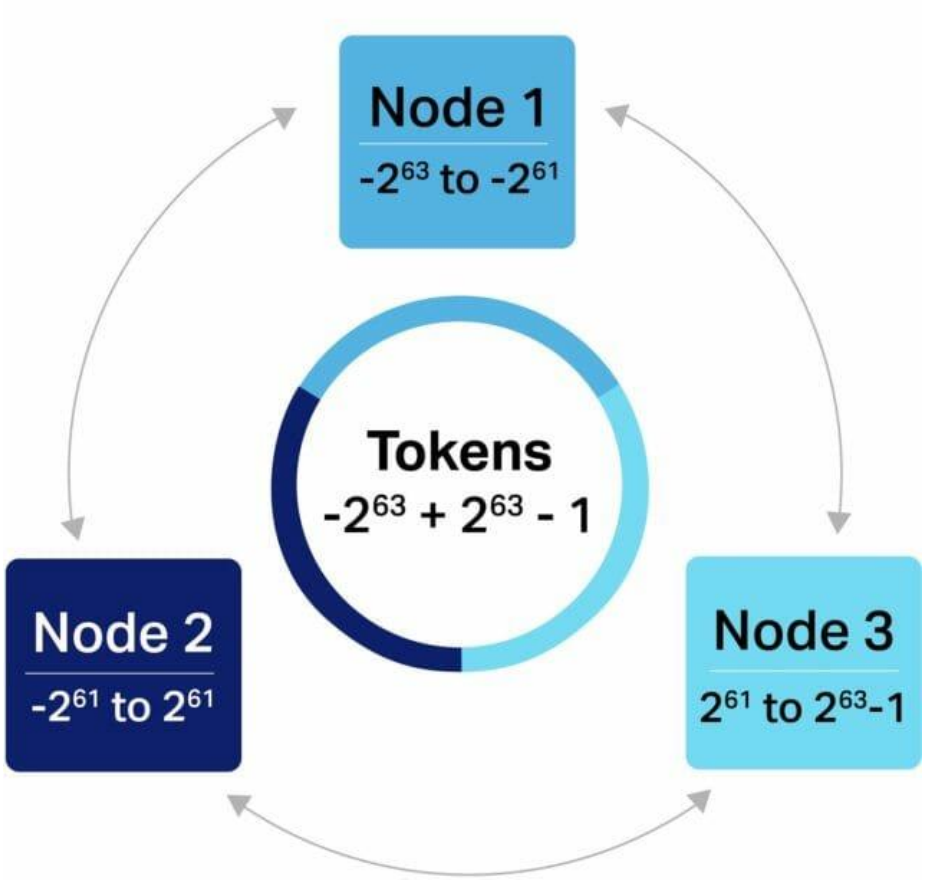
VERSIÓN 1.0

ALTA DISPONIBILIDAD
BIG TABLE

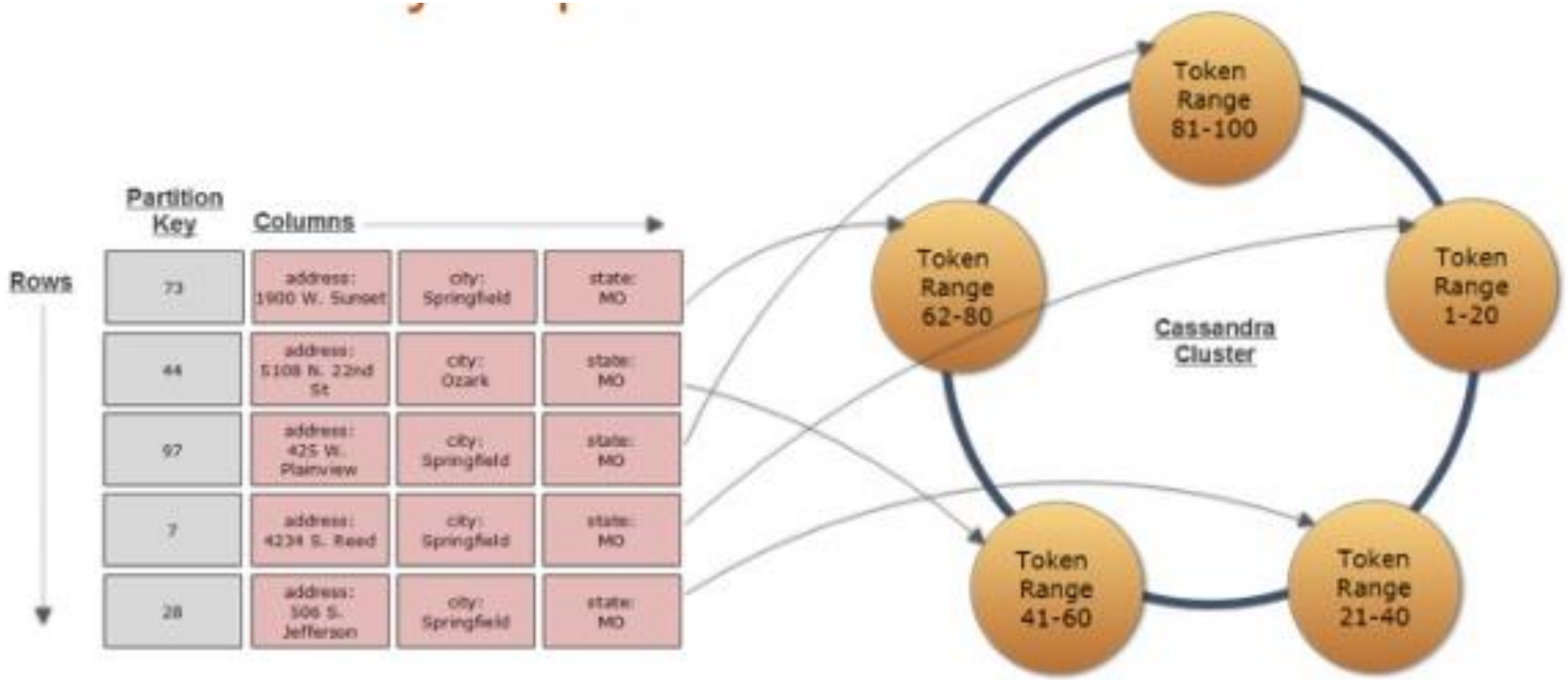




CASSANDRA USES 'TOKENS' (A LONG VALUE OUT OF RANGE -2^63 TO +2^63 -1) FOR DATA DISTRIBUTION AND INDEXING

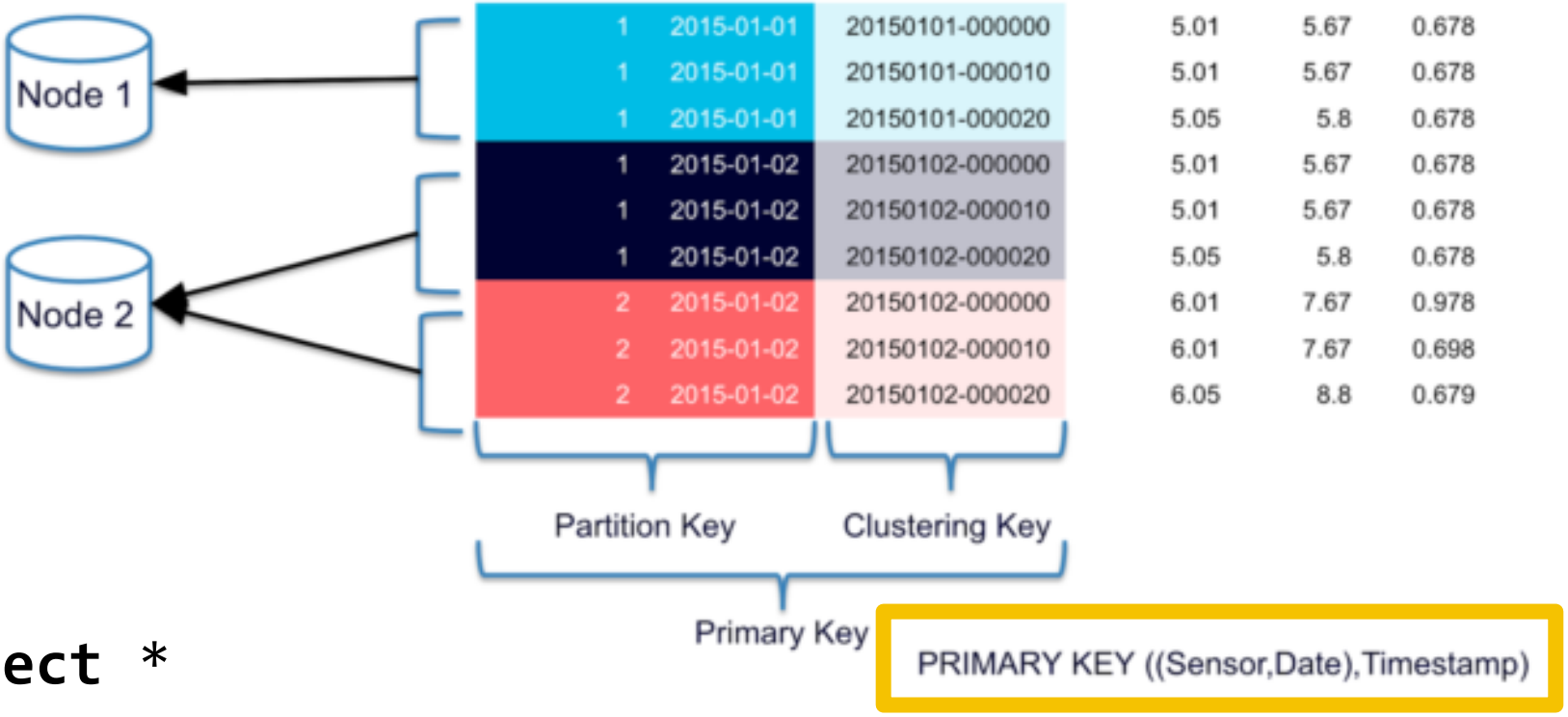


EJEMPLO DE DISTRIBUCIÓN INFO EN EL CLUSTER



PRIMARY KEY = PARTITION KEY + [CLUSTERING COLUMNS]

HASH(‘1 2015-01-01’) -> Nodo1
HASH(‘1 2015-01-02’) -> Nodo2
HASH(‘2 2015-01-02’) -> Nodo2
HASH(‘3 2015-01-01’) -> Nodo1



Sensor #	Date	Timestamp	Metric1	Metric2	Metric3
1	2015-01-01	20150101-000000	5.01	5.67	0.678
1	2015-01-01	20150101-000010	5.01	5.67	0.678
1	2015-01-01	20150101-000020	5.05	5.8	0.678
1	2015-01-02	20150102-000000	5.01	5.67	0.678
1	2015-01-02	20150102-000010	5.01	5.67	0.678
1	2015-01-02	20150102-000020	5.05	5.8	0.678
2	2015-01-02	20150102-000000	6.01	7.67	0.978
2	2015-01-02	20150102-000010	6.01	7.67	0.698
2	2015-01-02	20150102-000020	6.05	8.8	0.679

```
select *
from t
where sensor = 3 and date = ‘2015-01-01’
      and Timestamp >= ‘20150101-000010’;
```

CLAVE PRIMARIA
SOLO CLAVE DE PARTICIÓN

```
CREATE TABLE t (
    id int,
    k int,
    v text,
    PRIMARY KEY (id)
);
```

```
select *
from t
where id = 123;
```

```
select *
from t
where k = 456;
```

```
select *
from t
where v = 'abc';
```

```
select *
from t
where id > 123;
```

```
select *
from t
where id = 123 and
      k = 456;
```

```
select *
from t
where id = 123
order by k;
```

```
select *
from t
where id > 123
order by k;
```

CLAVE PRIMARIA
PARTICIÓN Y CLUSTERIZACIÓN

```
CREATE TABLE t (
   id int,
   c text,
   k int,
   v text,
   PRIMARY KEY (id,c)
);
```

```
select *
from t
where id = 123;
```

```
select *
from t
where id = 123 and
      c = ‘abc’;
```

```
select *
from t
where id = 123 and
      c >= ‘abc’;
```

```
select *
from t
where id = 123
order by  c asc;
```

CLAVE PRIMARIA

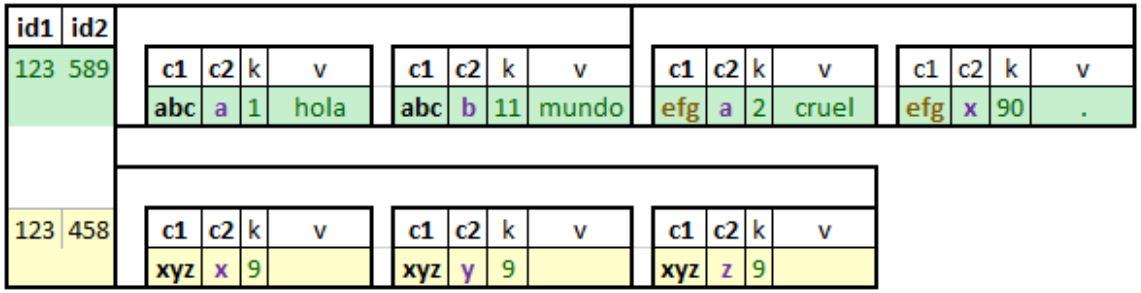
VARIAS CLAVE DE PARTICIÓN Y CLUSTERIZACIÓN

```
CREATE TABLE t (
   id1 int,
   id2 int,
   c1 text,
   c2 text
   k int,
   v text,
   PRIMARY KEY ((id1,id2),c1,c2)
);
```

id1	id2	c1	c2	k	v
123	589	abc	a	1	hola
		abc	b	11	mundo
		efg	a	2	cruel
		efg	x	90	.
123	458	xyz	x	9	
		xyz	y	9	
		xyz	z	9	

CLAVE PRIMARIA (1/3)
VARIAS CLAVE DE PARTICIÓN Y CLUSTERIZACIÓN

```
CREATE TABLE t (
   id1 int,
   id2 int,
   c1 text,
   c2 text
   k int,
   v text,
   PRIMARY KEY ((id1,id2),c1,c2)
);
```




```
select *
from t
where id1 = 123 and
      id2 = 589;
```

```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 > 'abc';
```

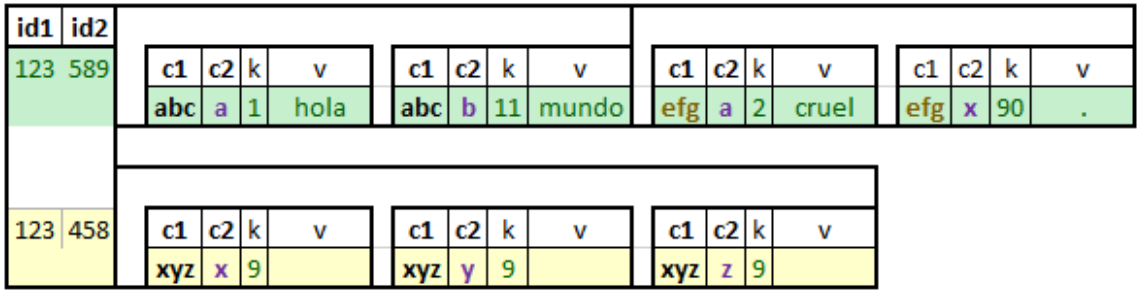
```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 = 'abc';
```

OK

```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 > 'abc'
order by c2;
```

CLAVE PRIMARIA (2/3)
VARIAS CLAVE DE PARTICIÓN Y CLUSTERIZACIÓN

```
CREATE TABLE t (
   id1 int,
   id2 int,
   c1 text,
   c2 text
   k int,
   v text,
   PRIMARY KEY ((id1,id2),c1,c2)
);
```




```
select *
from t
where id1 = 123 and
      id2 >= 589;
```

```
select *
from t
where id1 = 123 and
      id2 = 589 and
order by c2;
```

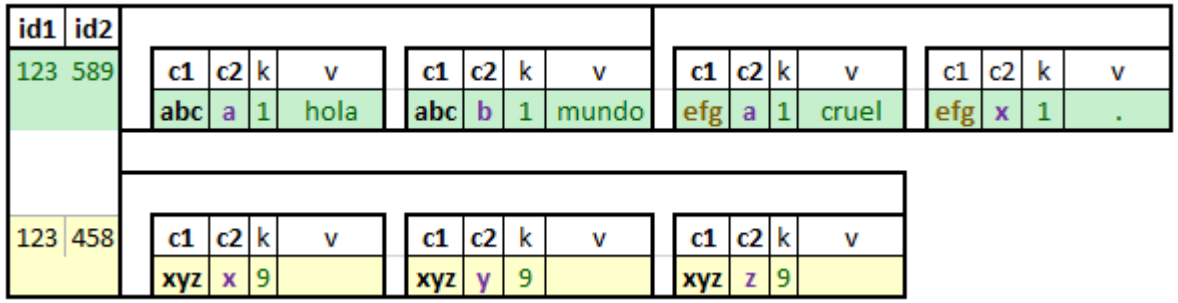
```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c2 = 'a';
```

```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 > 'abc' and
      c2 >= 'b';
```

ERROR!

CLAVE PRIMARIA (3/3)
VARIAS CLAVE DE PARTICIÓN Y CLUSTERIZACIÓN

```
CREATE TABLE t (
   id1 int,
   id2 int,
   c1 text,
   c2 text
   k int,
   v text,
   PRIMARY KEY ((id1,id2),c1,c2)
);
```

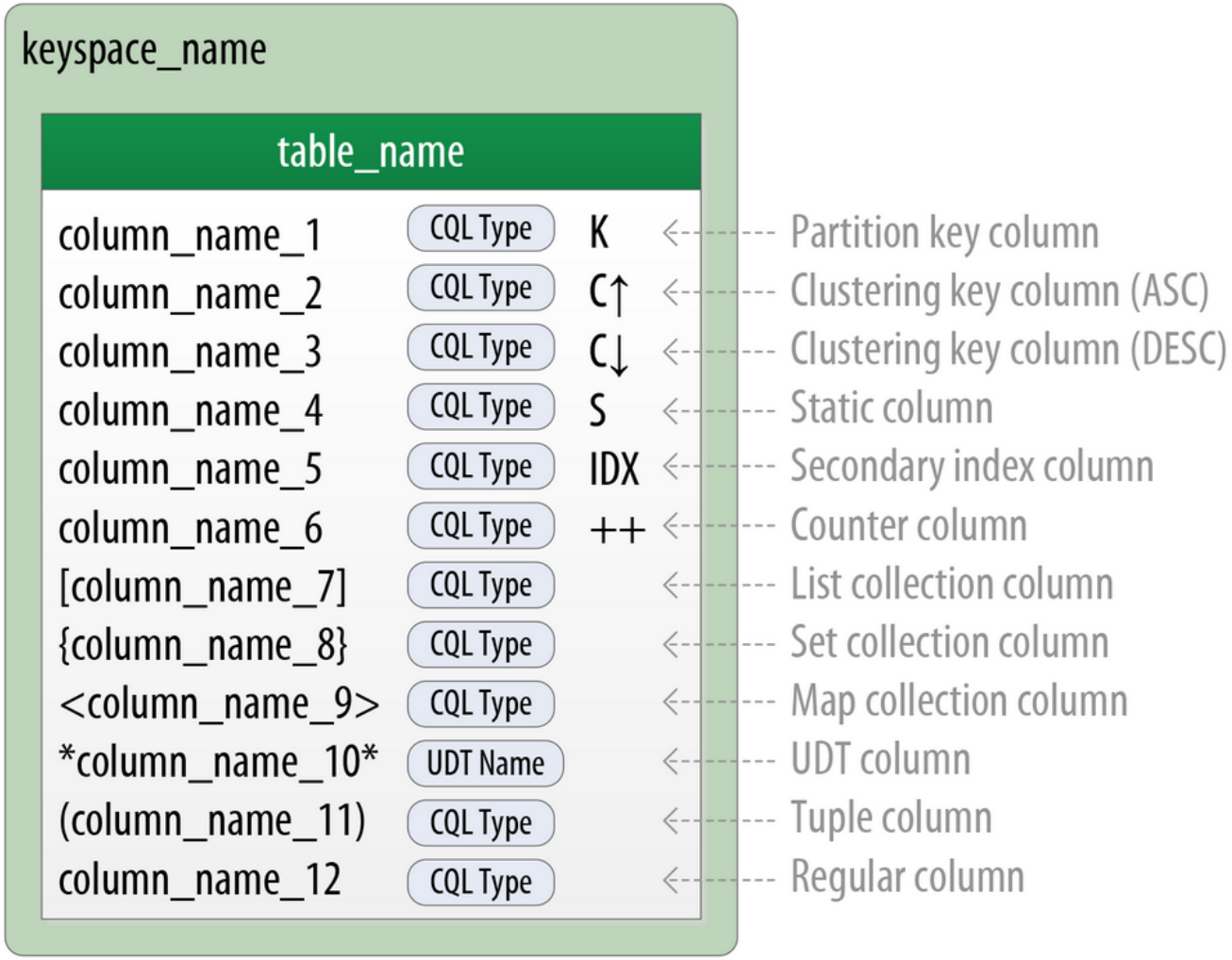



```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 = 'abc' and
      c2 >= 'b';
```

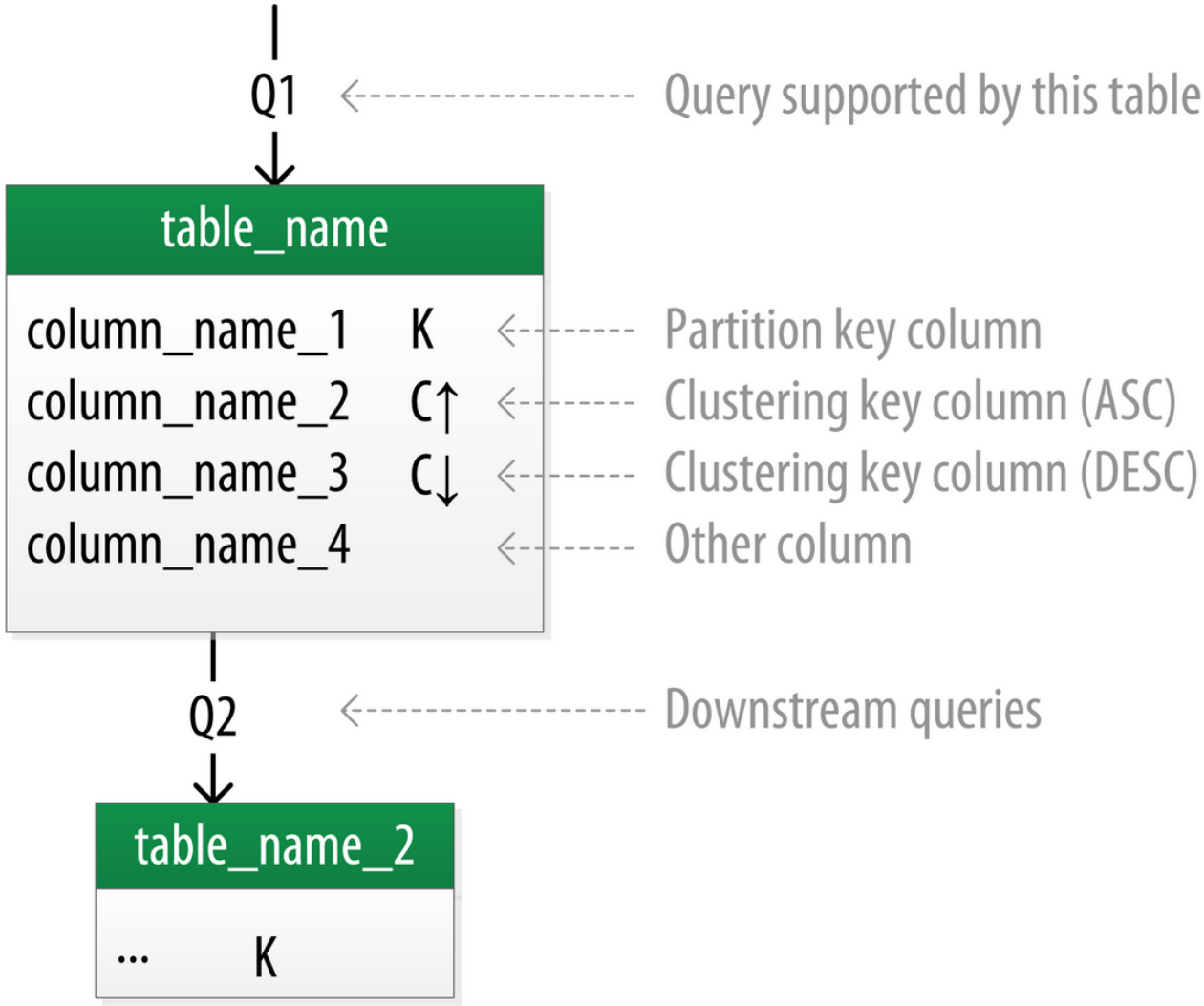
```
select *
from t
where id1 = 123 and
      id2 = 589 and
      c1 = 'abc' and
      c2 >= 'b'
order by c1,c2;
```

OK

ESTANDAR

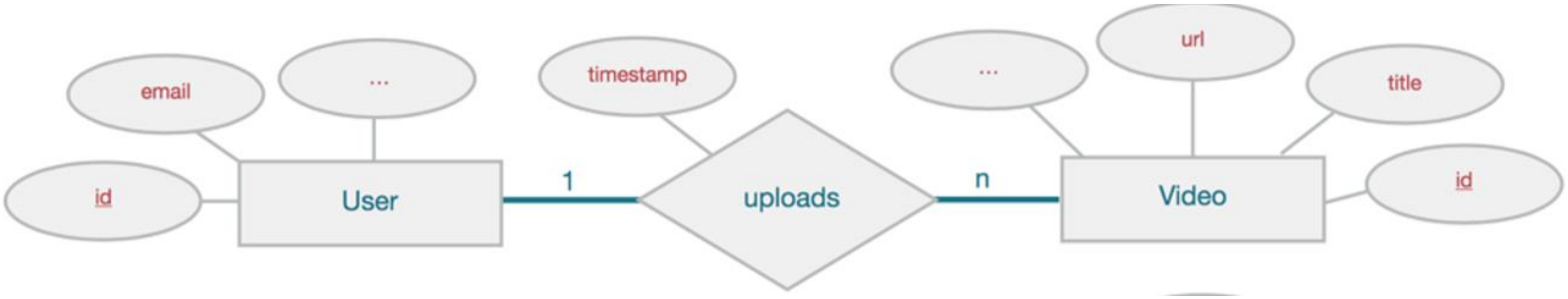

keyspace_name
table_name
column_name_1
CQL Type
K
Partition key column
column_name_2
CQL Type
C↑
Clustering key column (ASC)
column_name_3
CQL Type
C↓
Clustering key column (DESC)
column_name_4
CQL Type
S
Static column
column_name_5
CQL Type
IDX
Secondary index column
column_name_6
CQL Type
++
Counter column
[column_name_7]
CQL Type
List collection column
{column_name_8}
CQL Type
Set collection column
<column_name_9>
CQL Type
Map collection column
column_name_10
UDT Name
UDT column
(column_name_11)
CQL Type
Tuple column
column_name_12
CQL Type
Regular column

TABLA

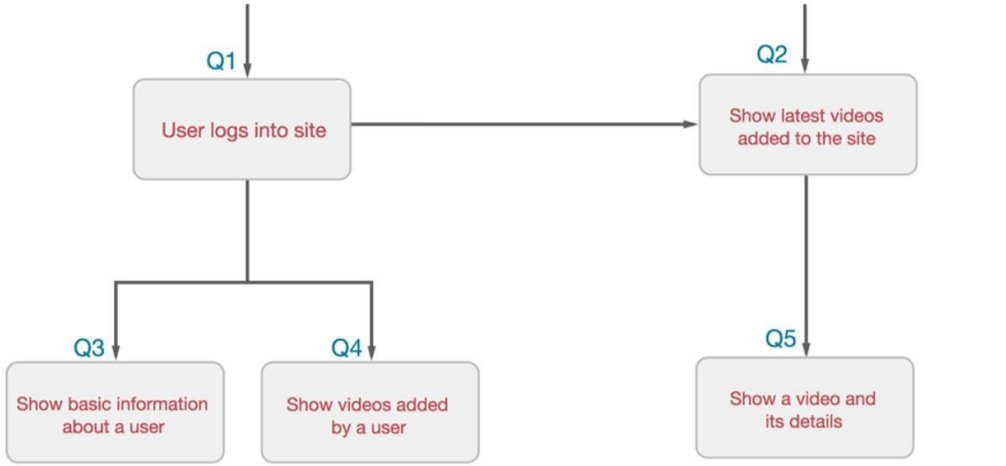


INSUMOS DISEÑO CASSANDRA

Modelo ER



WorkFlow



ACCESS PATTERNS

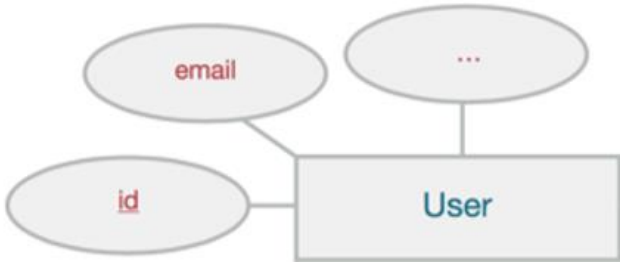
Q1: Find a user with a **specified email**
Q2: Find most recently uploaded **videos**
Q3: Find a **user with a specified id**
*Q4: Find videos uploaded by a **user with a known id** (in a time range and order by video id)
Q5: Find a video with a **specified video id**

MODELO ER CON WORKFLOW

Q1: Buscar un usuario con un email especifico
Recuperar usuario id y clave

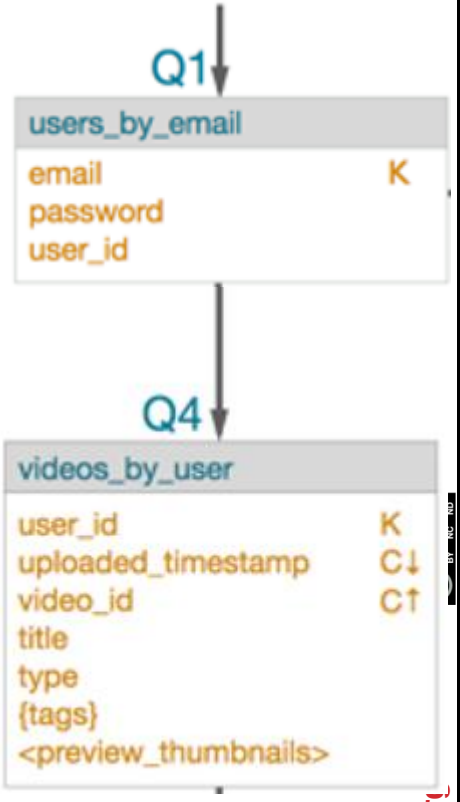
Given

email =

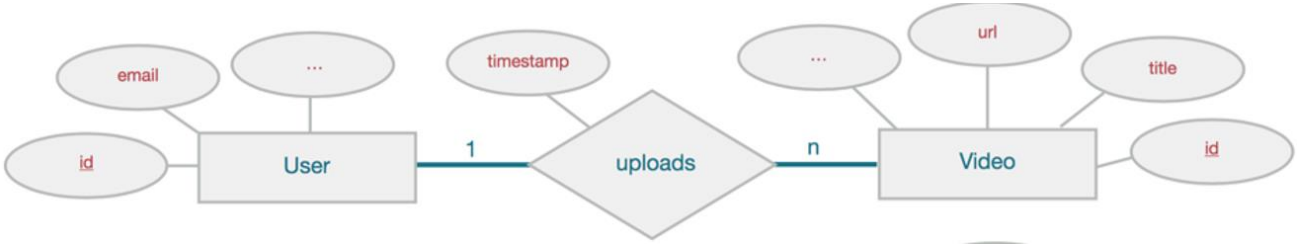


Find

Id (user_id)
password



Q4: Buscar los videos de un usuario en particular en un rango de tiempo.
Desplegando la información ordenada por el id del video.



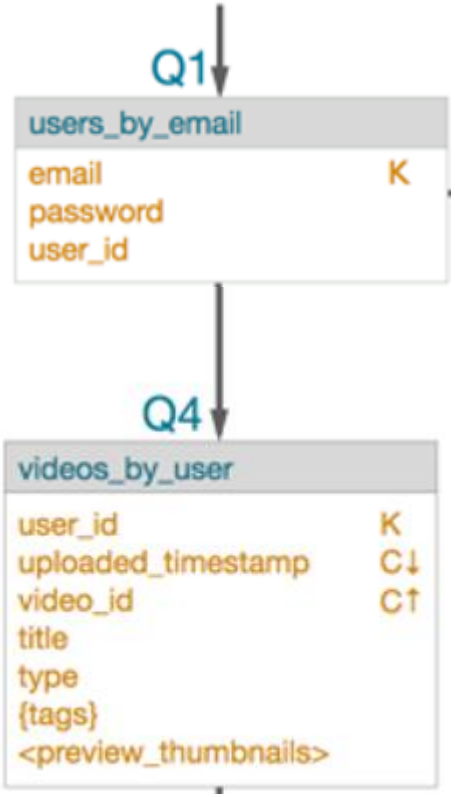
Given

user_id **(igualdad)**
uploaded_timestamp **(rango)**

Find

video_id **(ordenar asc)**
title
type
{tags}

TRABAJANDO CON WORKFLOW




```
select password, user_id
from users_by_email
where email = 'sergalpe@gmail.com';
```

```
select title, type, tags, video_id
from videos_by_user
where
user_id = '550e8400-e29b-41d4-a716-446655440000' and
uploaded_timestamp <= 1597257136000
order by video_id;
```

FIN

sergalpe@gmail.com