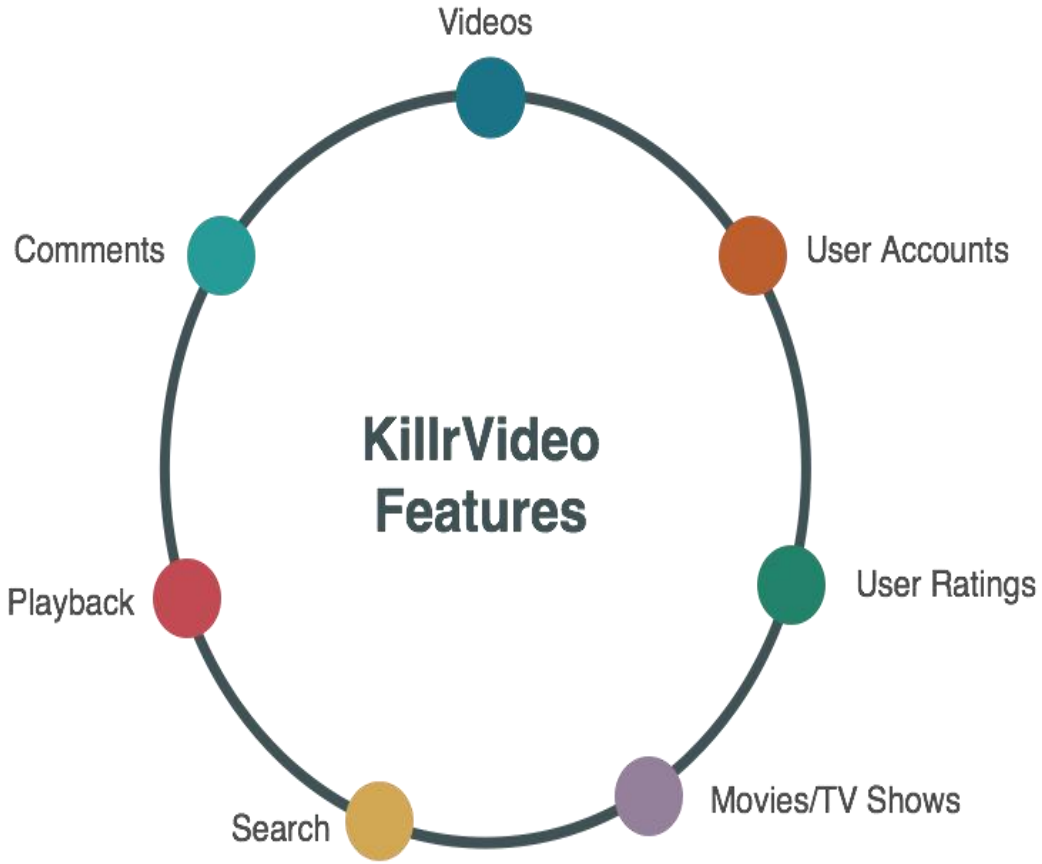
• LAB

• **EXERCISES ADAPTED FROM DATASTAX TRAINING**

Application- KillrVideo

- **Objectifs:**
- Passage à l’échelle – plus d’utilisateurs et de vidéos
- Disponibilité
- Facilité de gestion et



Create the keyspace

- CREATE KEYSPACE demoVideo
- WITH REPLICATION = {
- 'class': 'SimpleStrategy',
- 'replication_factor' : 1
- };

- USE demoVideo;

- Note: SOURCE './myscript.cql';

Example introduction-creation of the table

- CREATE TABLE videos
- (id int,
- name text,
- runtime int,
- year int,
- PRIMARY KEY (id));

insertion

- Insert this data into a video table
- Either directly with the INSERT clause or by using a CSV file and the COPY clause

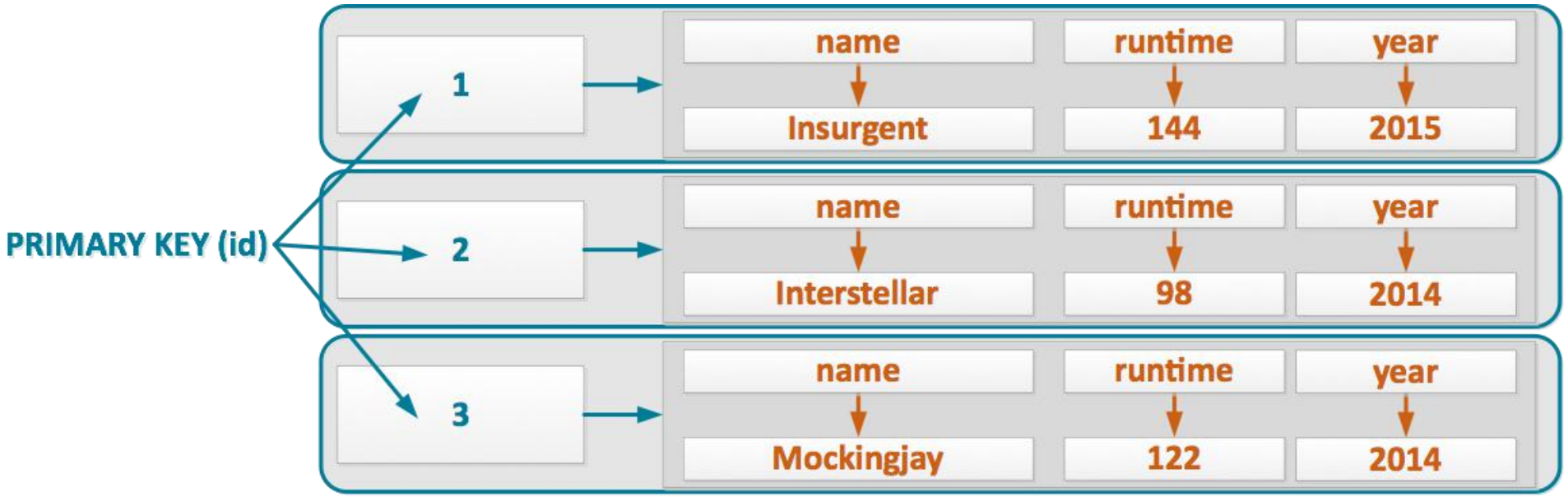
id	name	runtime	year
1	Insurgent	119	2015
2	Interstellar	98	2014
3	Mockingjay	122	2014

querying

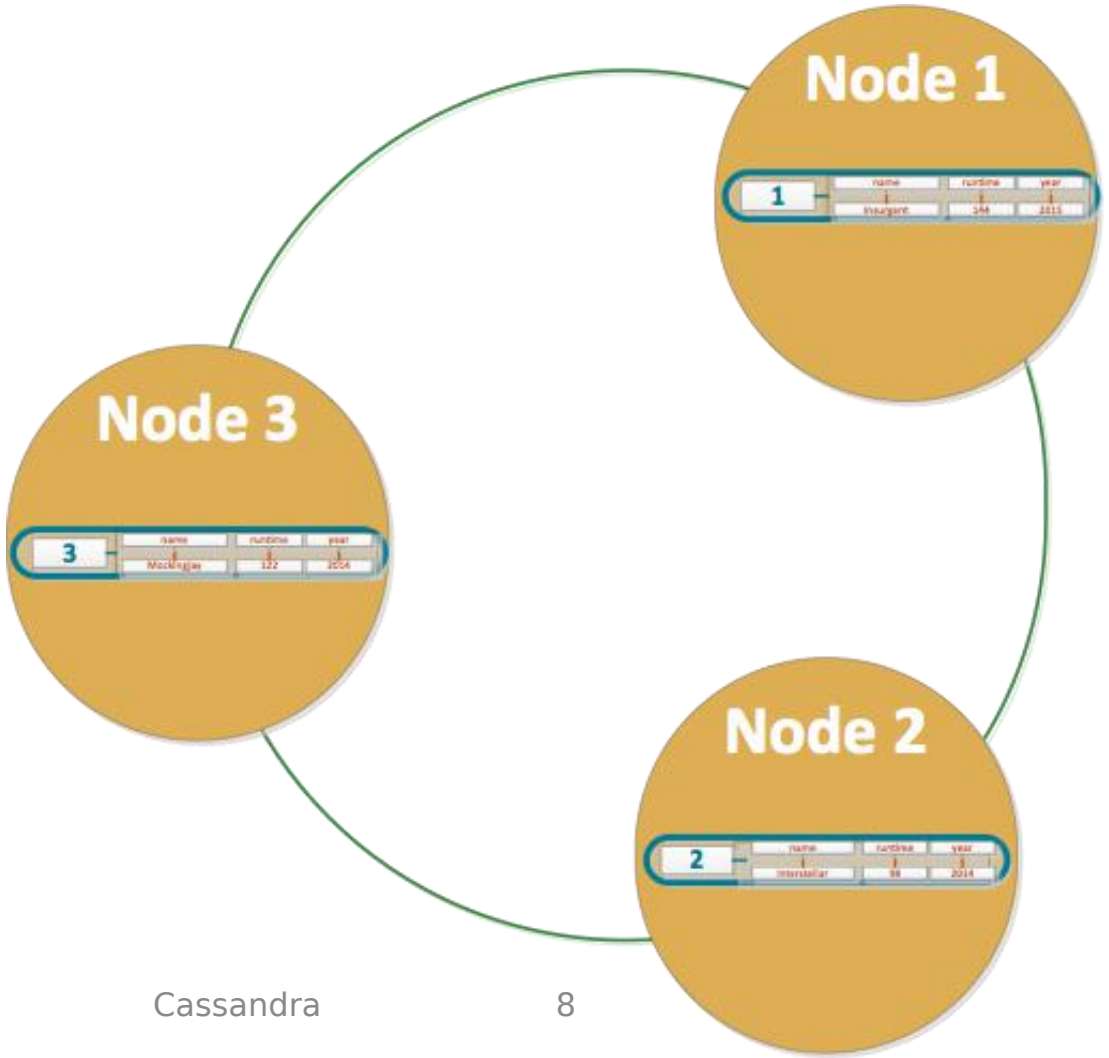
- How many rows have been inserted
- View All Records
- Show information about the video «insurgent»
- Show videos with year greater than 2014

  What do you get? Why?

Physical storage



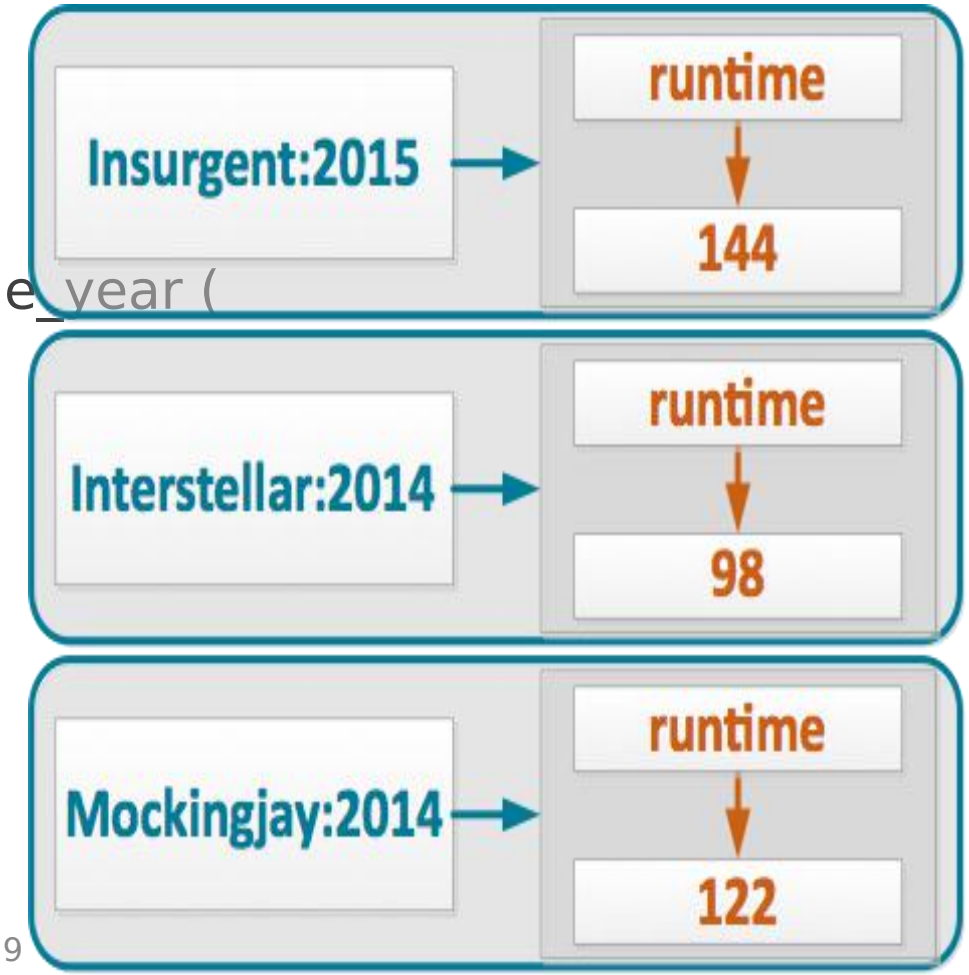
Partitioned storage



Is this a solution? Try it

-

- CREATE TABLE videos_by_name_year (
- name text,
- runtime int,
- year int,
- PRIMARY KEY ((name, year)));



queries

- Find the movie "Insurgent" made in 2015
- Find information about the film "Interstellar »
- What are the films made in 2014?

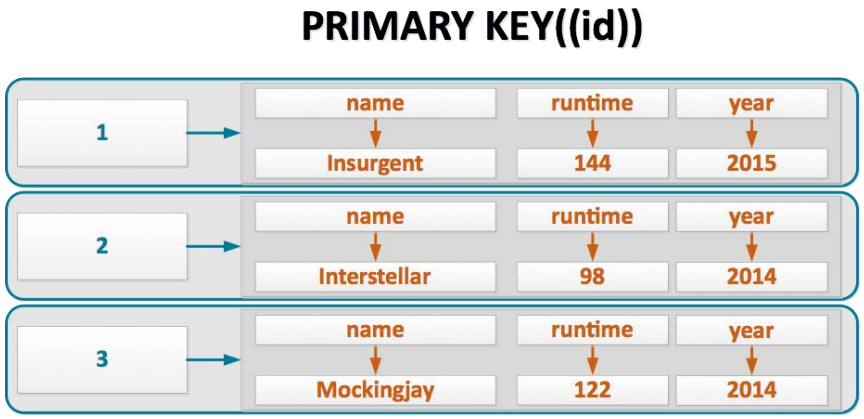
Cassandra-UPSERTS

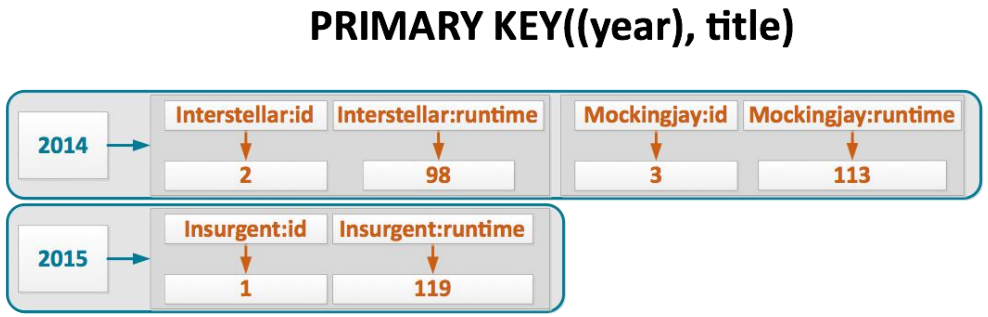
- INSERT INTO videos_by_name_year (name , year , runtime) VALUES ('Insurgent',2015, 127) ;
-
- SELECT count(*) from videos_by_name_year

- What’s happen?

Clustering columns

- CREATE TABLE videos_by_year (
- id int,
- name text,
- runtime int,
- year int,
- PRIMARY KEY ((year), name));





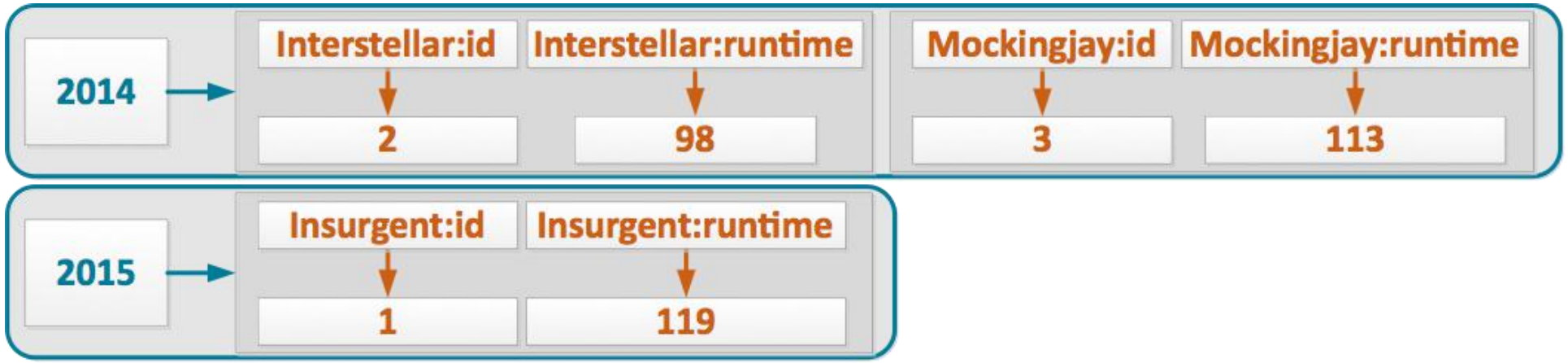
Clustering column with order

- Default ascending order
  If we want to specify a descending order:

- CREATE TABLE videos_by_year (
- id int,
- name text,
- runtime int,
- year int,
- PRIMARY KEY ((year), name))
- WITH CLUSTERING ORDER BY (name DESC);

Querying clustering Columns

- SELECT * FROM videos_by_year WHERE year = 2014 AND name = 'Mockingjay';
- Ou (opérations de comparaison)
- SELECT * FROM videos_by_year WHERE year = 2014 AND name >= 'Interstellar';



Alter table

- ALTER TABLE table1 ADD another_column text;
- ALTER TABLE table1 DROP another_column;
- - The column PRIMARY KEY can’t be modified
- - Delete data
- TRUNCATE table1;

Multi-valued column

- A column can contain several values (unlike RDBMS)

- SET <TEXT> collection of typed and ordered values (depending on value)
- LIST <TEXT> ordered by position
- MAP <TEXT, INT> key-value collection ordered by key

UDT (User defined type)

- CREATE TYPE address (
- street text,
- city text,
- zip_code int,
- phones set<text>
-);

- CREATE TYPE full_name (
- first_name text,
- last_name text
-);

Alter table videos

- Add a column tags (that can contain multiple tag values)
- Add some tags to your csv file or directly to records inserted into videos

UDT

- Create a video_encoding UDT following the example
- {encoding: '1080p', height: 1080, width: 1920, bit_rates: {'3000 Kbps', '4500 Kbps', '6000 Kbps'}

Field Name	Data Type
encoding	text
height	int
width	int
bit_rates	set<text>

- Create a video_encoding.csv file containing video_id and the encoding information
Example:
- 1,"{encoding: '1080p', height: 1080, width: 1920, bit_rates: {'3000 Kbps', '4500 Kbps', '6000 Kbps'}}"

Alter table and add info

- Add a new encoding column to the videos table
- Insert new information from previously created videos_encoding.csv
- Show videos content

Counter

- Create a new table with a counter to update the number of videos for each tag and year

- CREATE TABLE videos_count_by_tag (
- tag TEXT,
- added_year INT,
- video_count counter,
- PRIMARY KEY (tag, added_year)
-);

Counter

- To update the counter: (launch some updates on the table)

- UPDATE videos_count_by_tag SET video_count = video_count + 1 WHERE tag=‘MyTag’ AND added_year=2015;
- Display the result

- Try a counter update with a tag and a year that does not exist in your table. What do you get?