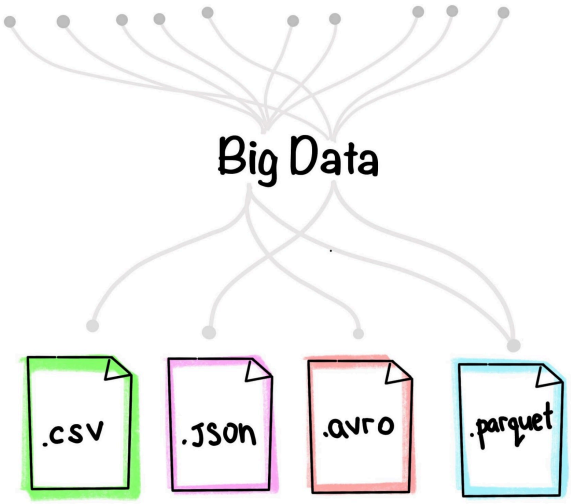
Minería de Datos II

Clase 7 - Formatos de Almacenamiento



La elección de un formato correcto puede traducirse en mejoras de performance y reducción de costos.

Un ejemplo conocido es el uso de Apache Parquet en AWS Athena.

[AWS Athena - Columnar Storage]

-Factores de Elección

Al momento de elegir un formato de almacenamiento, debemos considerar los siguientes puntos:

- ROW vs COLUMN: las consultas serán de tipo SELECT * o agregaciones AVG, SUM, etc

	day	location	product	sale
row 1	2017-01-01	l1	p1	300
row 2	2017-01-01	l1	p2	40
row 3	2017-01-01	l2	p1	44
row 4	2017-02-01	l1	p1	200

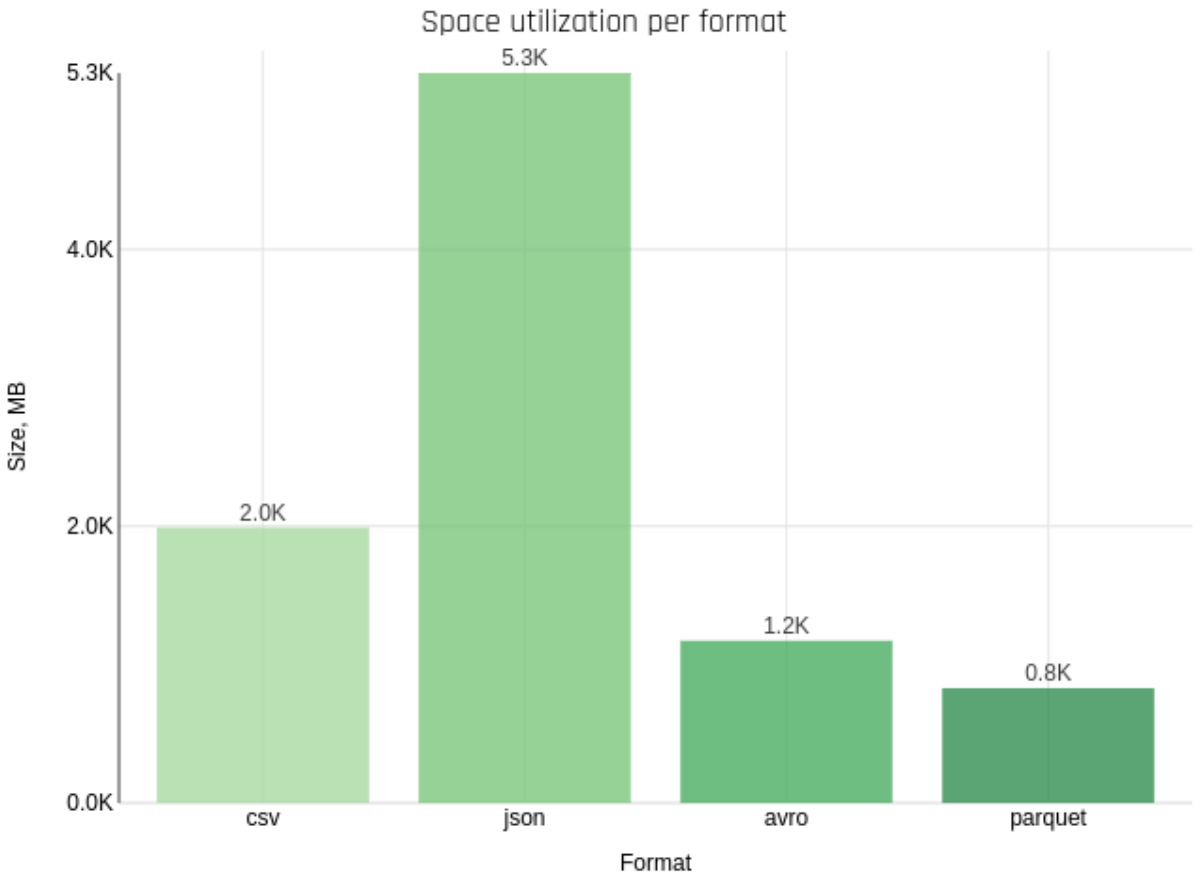
Traditional Memory Buffer	
row 1	2017-01-01
	l1
	p1
	300
row 2	2017-01-01
	l1
	p2
	40
row 3	2017-01-01
	l2
	p1
	44

Columnar Storage	
day	2017-01-01
	2017-01-01
	2017-01-01
	2017-01-02
location	l1
	l1
	l2
	l1
product	p1
	p2
	p1
	p1

- SCHEMA EVOLUTION: que sucede si debemos agregar, eliminar o modificar un campo

```
{"namespace": "drwho.avro",
 "type": "record",
 "name": "drwho",
 "fields": [
    {"name": "drwho_season", "type": ["null","string"], "aliases": ["doctor_who_season"]},
    {"name": "drwho_actor", "type": ["null","string"], "aliases": ["doctor_actor"]},
    {"name": "episode_no", "type": ["null","string"]}, {"name": "episode_title", "type": ["null","string"]},
    {"name": "date_from", "type": ["null","string"]},  {"name": "date_to", "type": ["null","string"]},
    {"name": "estimated", "type": "string"}, {"name": "planet", "type": ["null","string"]},
    {"name": "sub_location", "type": ["null","string"]}, {"name": "main_location", "type": ["null","string"]},
    {"name": "hd", "type": "string", "default": "no"}
]}
```

- COMPRESSION: equilibrio entre espacio en disco utilizado y tiempo de procesamiento



Enlace sugerido: https://luminousmen.com/post/big-data-file-formats

-CSV

austin_bikeshare_stations.csv

```
latitude,longitude,name,station_id,status
30.27041,-97.75046,West & 6th St.,2537,active
30.26452,-97.7712,Barton Springs Pool,2572,active
30.27595,-97.74739,ACC - Rio Grande & 12th,2545,closed
30.2848,-97.72756,Red River & LBJ Library,1004,closed
30.26694,-97.74939,Nueces @ 3rd,1008,moved
30.26751,-97.74802,Republic Square,2500,moved
30.24891,-97.75019,South Congress & Elizabeth,2504,active
30.26461,-97.73049,Waller & 6th St.,2536,closed
30.26217,-97.72743,Plaza Saltillo,2542,active
30.28576,-97.74181,UT West Mall @ Guadalupe,2548,active
```

Texto plano delimitado por el carácter de la coma (pueden utilizarse otros separadores). Generalmente este formato lo utilizan aplicaciones tradicionales para exportar datos hacia otros sistemas.

-jSON

```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

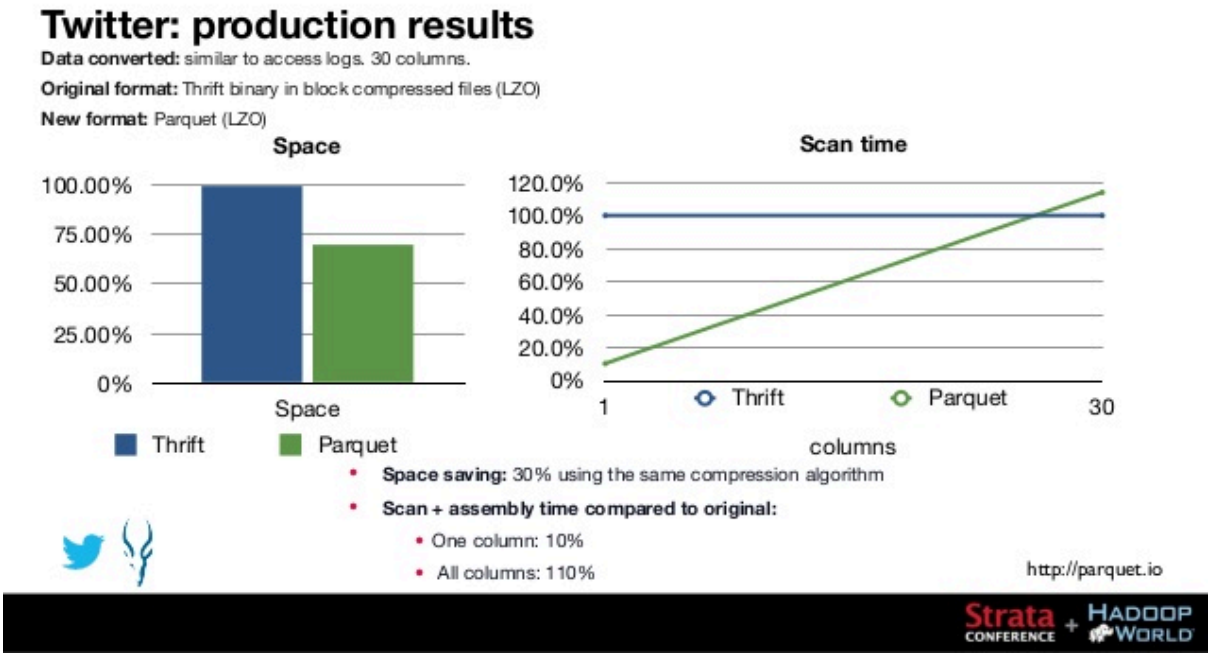
Permite representar estructuras jerárquicas y relaciones en un solo documento (ejemplo MongoDB).
Es el formato estándar para comunicaciones vía HTTP (ejemplo lectura de datos de Twitter)

-Avro

Almacena los datos en formato binario para reducir el tamaño y mejorar la performance.
La definición de los datos (schema) se almacena en formato JSON.
Es recomendable utilizarlo para consultas de tipo SELECT *.

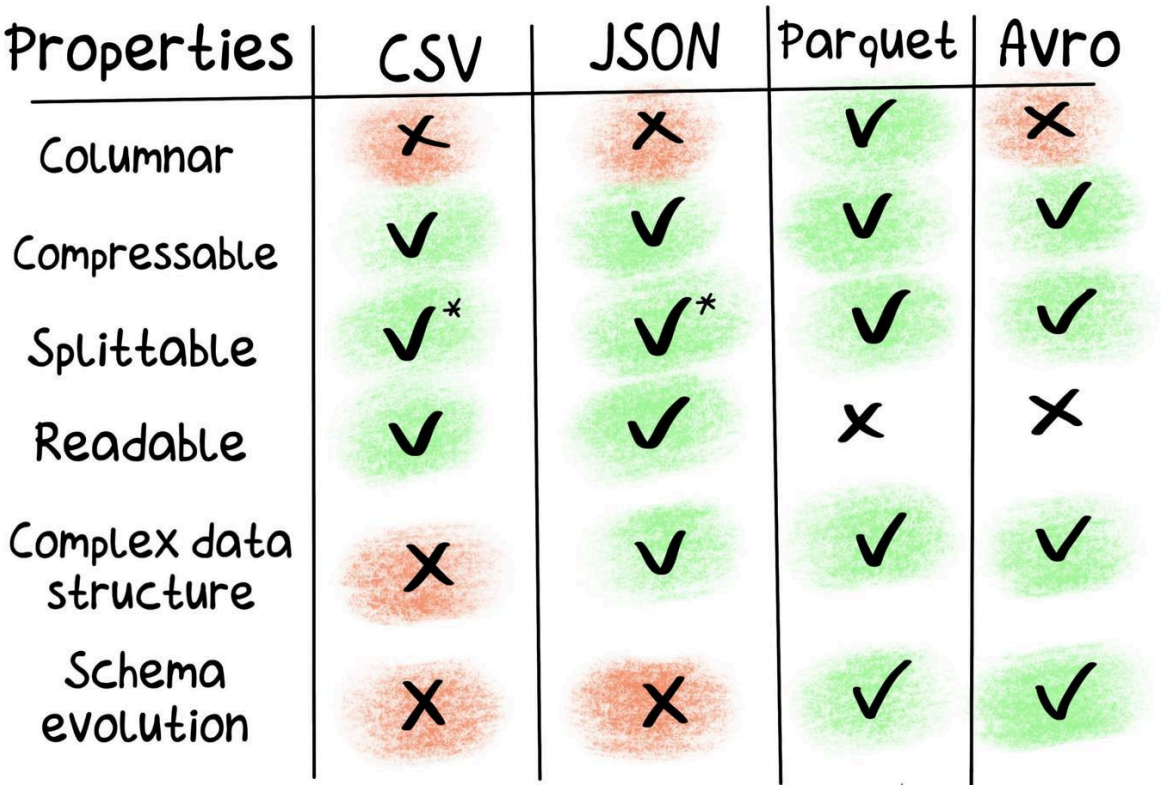
-Parquet

Es un formato de almacenamiento columnar que surge de la colaboración de Twitter y Cloudera.
Los datos se almacenan en formato binario y al final del archivo se agrega la metadata (schema).
Este formato es ideal para agregaciones AVG, SUM, etc.



https://www.slideshare.net/julienledem/parquet-stratany-hadoopworld2013

-Resumen:



Enlaces sugeridos:

- Parquet https://parquet.apache.org/documentation/latest/
- Avro https://avro.apache.org/docs/current/
- JSON https://www.json.org/json-es.html

-Hadoop Compression Types

- gzip - org.apache.hadoop.io.compress.GzipCodec
- bzip2 - org.apache.hadoop.io.compress.BZip2Codec
- LZO - com.hadoop.compression.lzo.LzopCodec
- Snappy - org.apache.hadoop.io.compress.SnappyCodec
- Deflate - org.apache.hadoop.io.compress.DeflateCodec

https://docs.cloudera.com/documentation/enterprise/5-9-x/topics/introduction_compression.html

-Hive SerDes

Acrónimo de Serializer/Deserializer. Permite interpretar diferentes formatos.

SerDes disponibles en Hive

- Avro (Hive 0.9.1 and later)
- ORC (Hive 0.11 and later)
- RegEx
- Thrift
- Parquet (Hive 0.13 and later)
- CSV (Hive 0.14 and later)
- JsonSerDe (Hive 0.12 and later in hcatalog-core)

https://cwiki.apache.org/confluence/display/Hive/SerDe

-Ejemplo Parquet y Snappy

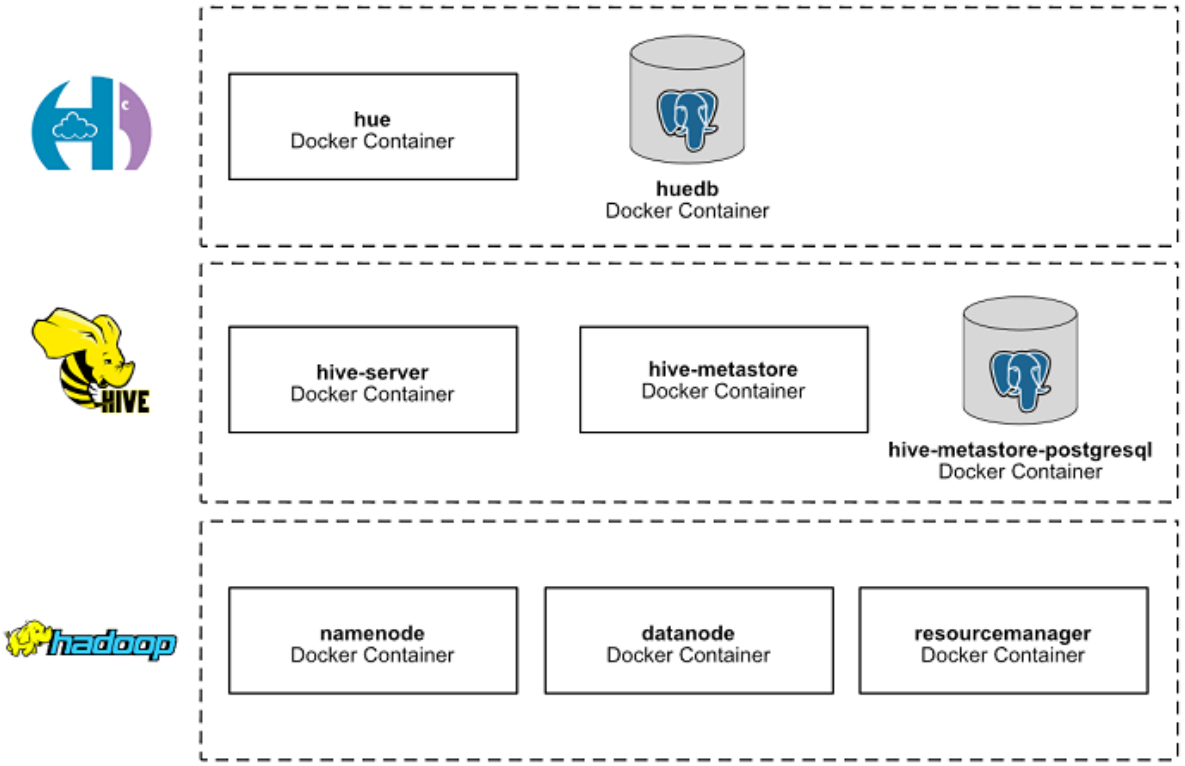
- In general LZO wins size benchmarks, Snappy good balance between size and CPU intensity.

```
led-zeppelin-albums.parquet/
•  _SUCCESS
•  _common_metadata
•  _metadata
•  Year=1969/
    – Part-r-00000-6d4d42e2-c13f-4bdf-917d-2152b24a0f24.snappy.parquet
    – Part-r-00001-6d4d42e2-c13f-4bdf-917d-2152b24a0f24.snappy.parquet
    – …
•  Year=1970/
    – Part-r-00000-35cb7ef4-6de6-4efa-9bc6-5286de520af7.snappy.parquet
    – ...
```

-Casos de Uso:

Laboratorio:

Se trabajará con el siguiente entorno de Docker Compose:



Instrucciones para su configuración:

- git clone https://github.com/tech4242/docker-hadoop-hive-parquet.git

- cd docker-hadoop-hive-parquet/
- sudo docker-compose up

Fuente: https://towardsdatascience.com/making-big-moves-in-big-data-with-hadoop-hive-parquet-hue-and-docker-320a52ca175

-Formatos de Almacenamiento en Hive

En la sección de archivos, cargar los archivos de la carpeta data y replicar la misma estructura de directorios en HDFS.

En la sección de mis documentos, cargar el archivo clase-04.json y luego seleccionar el editor Hive

Proyecto Integrador:

Hive

Se puede utilizar el entorno docker-compose-v2.yml

1) Crear tablas en Hive, a partir de los csv ingestados en HDFS.

Para esto, se puede ubicar dentro del contenedor correspondiente al servidor de Hive, y ejecutar desde allí los scripts necesarios

```
  sudo docker exec -it hive-server bash
  hive
```

2) Este proceso de creación de las tablas debe poder ejecutarse desde un shell script.

Nota: Para ejecutar un script de Hive, requiere el comando:
```
  hive -f <script.hql>
```

3) Las tablas creadas en el punto anterior a partir de archivos en formato csv, deben ser almacenadas en formato Parquet + Snappy. Tener en cuenta además de aplicar particiones para alguna de las tablas.