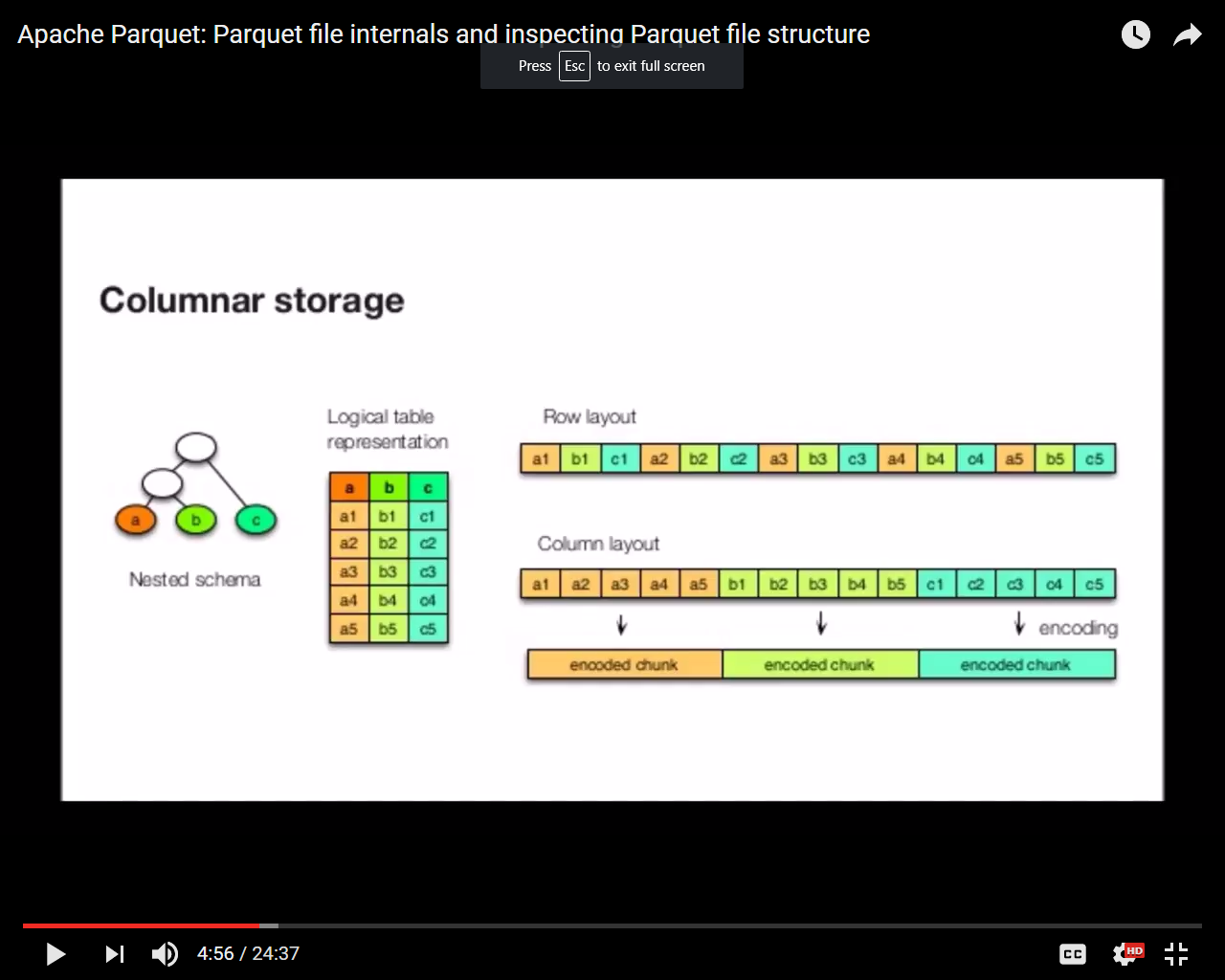
Parquet file format is columnar format



Parquet stores all columns together so that encoding is same type for each column set

To illustrate what columnar storage is all about, here is an example with three columns.



In a row-oriented storage, the data is laid out one row at a time as follows:



Whereas in a column-oriented storage, it is laid out one column at a time:

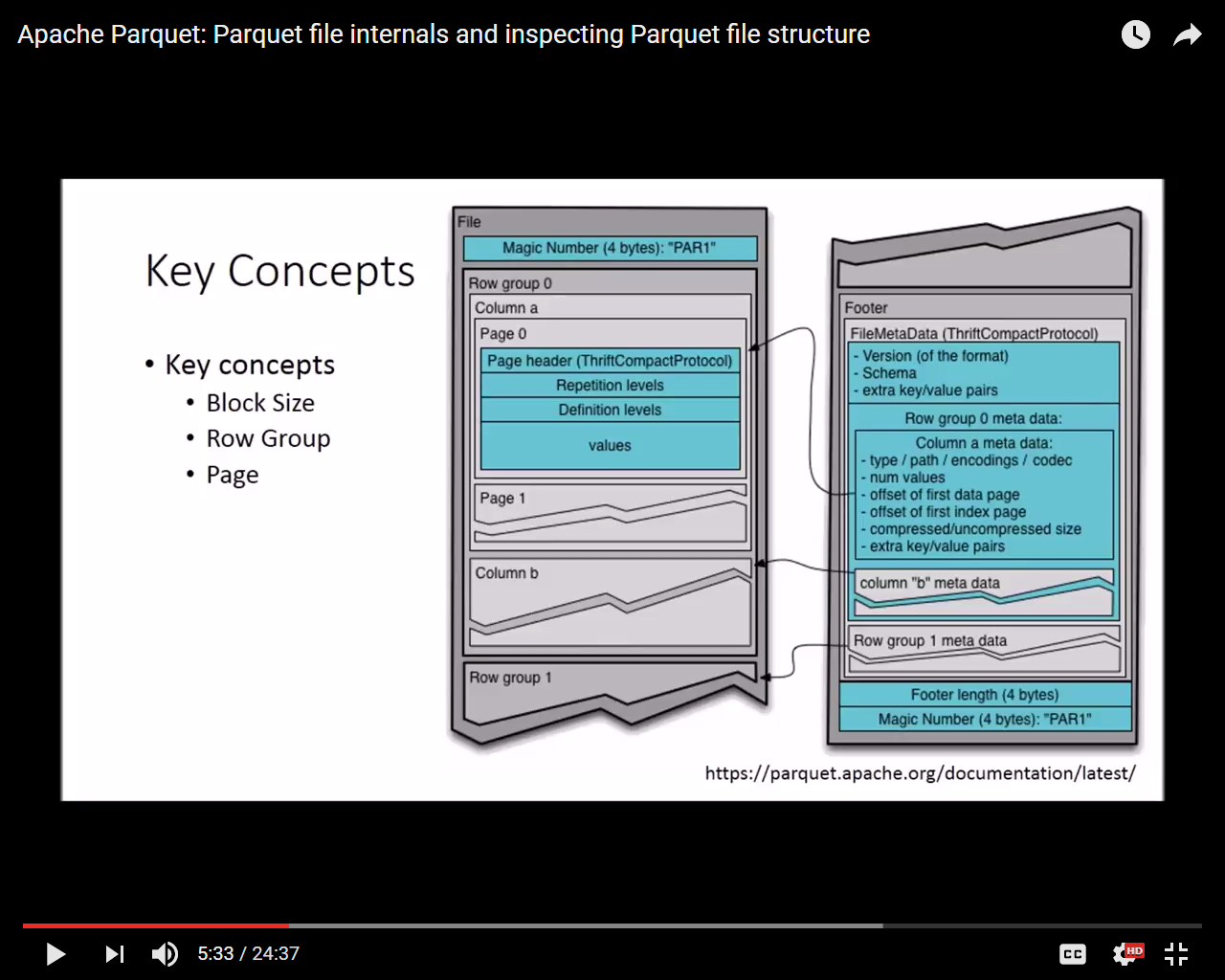


There are several advantages to columnar formats.

* Organizing by column allows for better compression, as data is more homogenous. The space savings are very noticeable at the scale of a Hadoop cluster.
* I/O will be reduced as we can efficiently scan only a subset of the columns while reading the data. Better compression also reduces the bandwidth required to read the input.
* As we store data of the same type in each column, we can use encodings better suited to the modern processors’ pipeline by making instruction branching more predictable.

Parquet support schema evolution, we can add new columns; modification of column is not supported as of now.

If we have millions of columns, there would be n number of parquet files which stores separate set of block as per configured parquet block size



* This block size is not same as hdfs block size, it should be multiple of hdfs block size , default parquet block size is 128 MB
* Each Block contains Row Group
* Each Row group contains set of columns , like in above diagram Row Group 0 contains Column a and Column B
* Each Column contains Page , it is lowest level which stores the details of data itself

If you set higher Row Group Size and higher Block size it improves overall read performance but It consume lots of memory when it comes to write performance

To get a rough idea of memory consumption we can multiply row group to total number of open files

We can use parquet tools to get details about parquet file , it default comes with Hadoop

To check schema

|  |
| --- |
| [root@mac55 ~]# **parquet-tools schema employee.parquet**  message spark\_schema {  optional int32 Empid;  optional binary EmpName (UTF8);  optional binary Dept (UTF8);  } |

To check data

|  |
| --- |
| [root@mac55 ~]# **parquet-tools head employee.parquet**  Empid = 1  EmpName = Shalaj  Dept = COE  Empid = 2  EmpName = Bhavesh  Dept = COE  Empid = 3  EmpName = Neeraj  Dept = COE  Empid = 4  EmpName = Ravi  Dept = COE  Empid = 5  EmpName = Chetan  Dept = COE |

To check the content of parquet file

|  |
| --- |
| [root@mac55 ~]# **parquet-tools meta employee.parquet**  creator: parquet-mr version 1.8.1 (build 4aba4dae7bb0d4edbcf7923ae1339f28fd3f7fcf)  extra: org.apache.spark.sql.parquet.row.metadata = {"type":"struct","fields":[{"name":"Empid","type":"int [more]...  file schema: spark\_schema  -------------------------------------------------------------------------------------------------------------------------  Empid: OPTIONAL INT32 R:0 D:1  EmpName: OPTIONAL BINARY O:UTF8 R:0 D:1  Dept: OPTIONAL BINARY O:UTF8 R:0 D:1  row group 1: RC:5 TS:211  -------------------------------------------------------------------------------------------------------------------------  Empid: INT32 SNAPPY DO:0 FPO:4 SZ:59/59/1.00 VC:5 ENC:PLAIN,BIT\_PACKED,RLE  EmpName: BINARY SNAPPY DO:0 FPO:63 SZ:94/93/0.99 VC:5 ENC:PLAIN,BIT\_PACKED,RLE  Dept: BINARY SNAPPY DO:0 FPO:157 SZ:63/59/0.94 VC:5 ENC:PLAIN\_DICTIONARY,BIT\_PACKED,RLE  [root@mac55 ~]# |

* We can get version from creator tag: parquet-mr **version 1.8.1**
* Extra tag gives you some more information , since we create this file using spark it shows spark related information
* File schema contains schema details
* Row group contains row group detail, Since our file is very small (just 5 records), it creates just one row group, we can also see compression algorithm (snappy), encoding(ENC:PLAIN,BIT\_PACKED,RLE)
* SZ:59/59/1.00 says original size/post compression size/compression ratio

Dump command

|  |
| --- |
| **[root@mac55 ~]# parquet-tools dump employee.parquet**  row group 0  -------------------------------------------------------------------------------------------------------------------------  Empid: INT32 SNAPPY DO:0 FPO:4 SZ:59/59/1.00 VC:5 ENC:BIT\_PACKED,PLAIN,RLE  EmpName: BINARY SNAPPY DO:0 FPO:63 SZ:94/93/0.99 VC:5 ENC:BIT\_PACKED,PLAIN,RLE  Dept: BINARY SNAPPY DO:0 FPO:157 SZ:63/59/0.94 VC:5 ENC:PLAIN\_DICTIONARY,BIT\_PACKED,RLE  Empid TV=5 RL=0 DL=1  ---------------------------------------------------------------------------------------------------------------------  page 0: DLE:RLE RLE:BIT\_PACKED VLE:PLAIN SZ:26 VC:5  EmpName TV=5 RL=0 DL=1  ---------------------------------------------------------------------------------------------------------------------  page 0: DLE:RLE RLE:BIT\_PACKED VLE:PLAIN SZ:55 VC:5  Dept TV=5 RL=0 DL=1 DS: 1 DE:PLAIN\_DICTIONARY  ---------------------------------------------------------------------------------------------------------------------  page 0: DLE:RLE RLE:BIT\_PACKED VLE:PLAIN\_DICTIONARY SZ:8 VC:5  INT32 Empid  -------------------------------------------------------------------------------------------------------------------------  \*\*\* row group 1 of 1, values 1 to 5 \*\*\*  value 1: R:0 D:1 **V:1**  value 2: R:0 D:1 **V:2**  value 3: R:0 D:1 V:3  value 4: R:0 D:1 V:4  value 5: R:0 D:1 V:5  BINARY EmpName  -------------------------------------------------------------------------------------------------------------------------  \*\*\* row group 1 of 1, values 1 to 5 \*\*\*  value 1: R:0 D:1 **V:Shalaj**  value 2: R:0 D:1 **V:Bhavesh**  value 3: R:0 D:1 V:Neeraj  value 4: R:0 D:1 V:Ravi  value 5: R:0 D:1 V:Chetan  BINARY Dept  -------------------------------------------------------------------------------------------------------------------------  \*\*\* row group 1 of 1, values 1 to 5 \*\*\*  value 1: R:0 D:1 **V:COE**  value 2: R:0 D:1 **V:COE**  value 3: R:0 D:1 V:COE  value 4: R:0 D:1 V:COE  value 5: R:0 D:1 V:COE |

Here we can see page level details

Also we can see that row group1 contain all three columns (EmpId,empname and dept) we can see value following by V:

So all columns are stored together