Looking under the hood of the:

Parquet Format

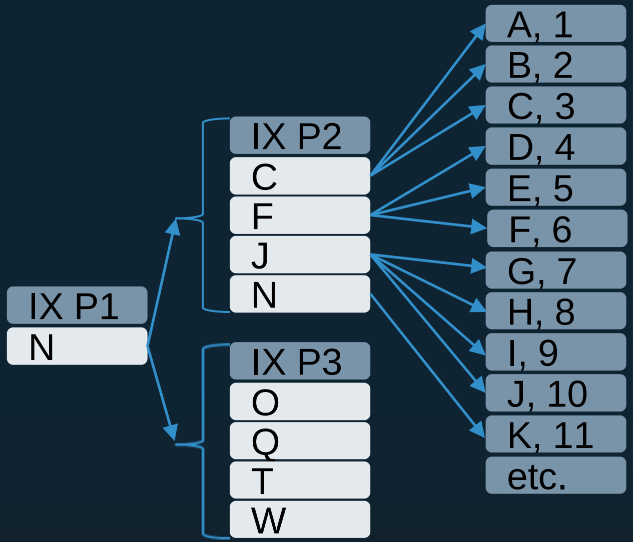
André Kamman

@andrekamman

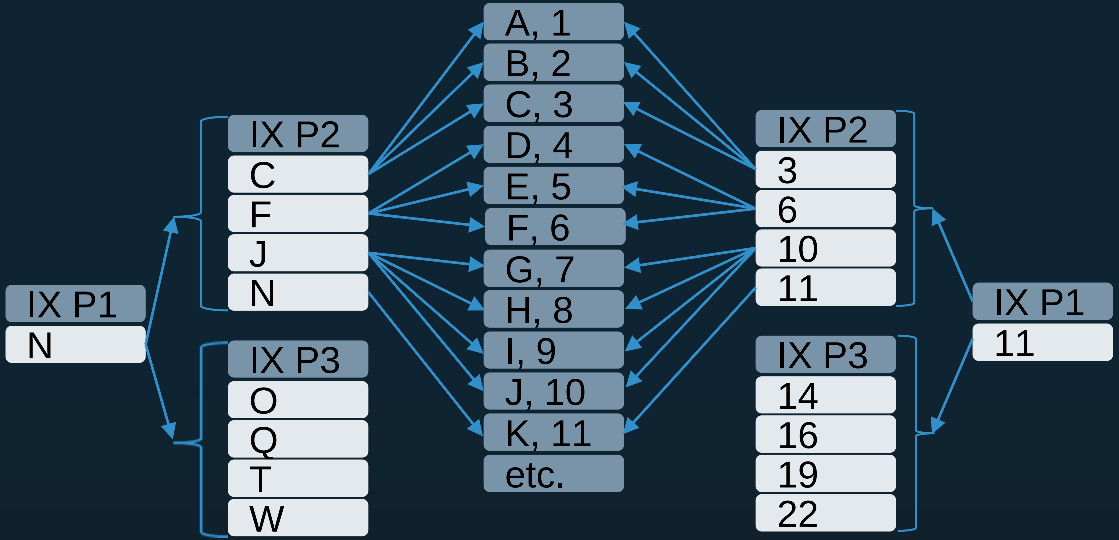
andrekamman@gmail.com

https://evals.datagrillen.com/

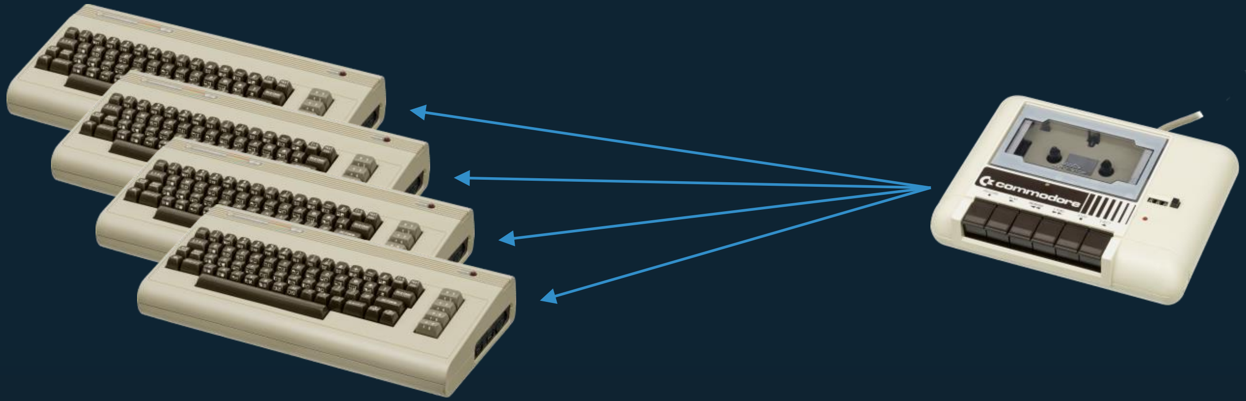
B-Tree

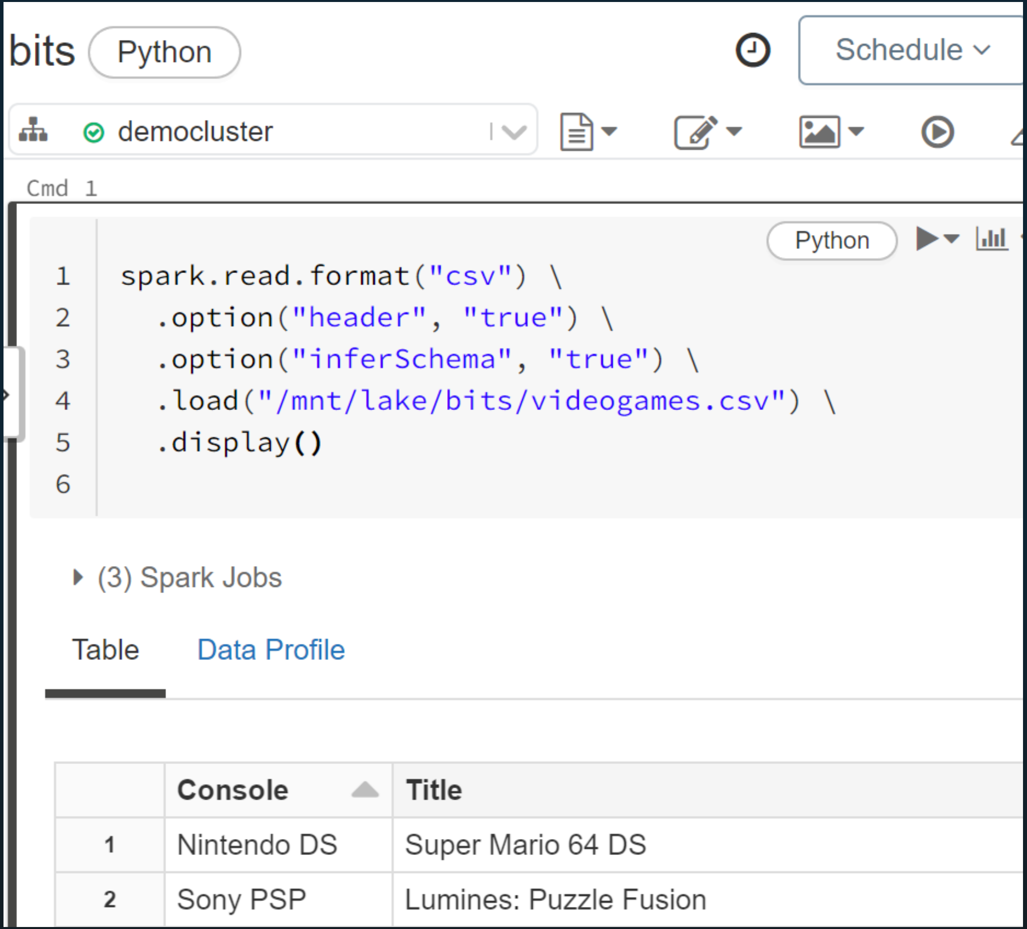


B-Tree


IX P1
N
IX P2
C
F
J
N
IX P3
O
Q
T
W
A, 1
B, 2
C, 3
D, 4
E, 5
F, 6
G, 7
H, 8
I, 9
J, 10
K, 11
etc.
IX P2
3
6
10
11
IX P3
14
16
19
22
IX P1
11

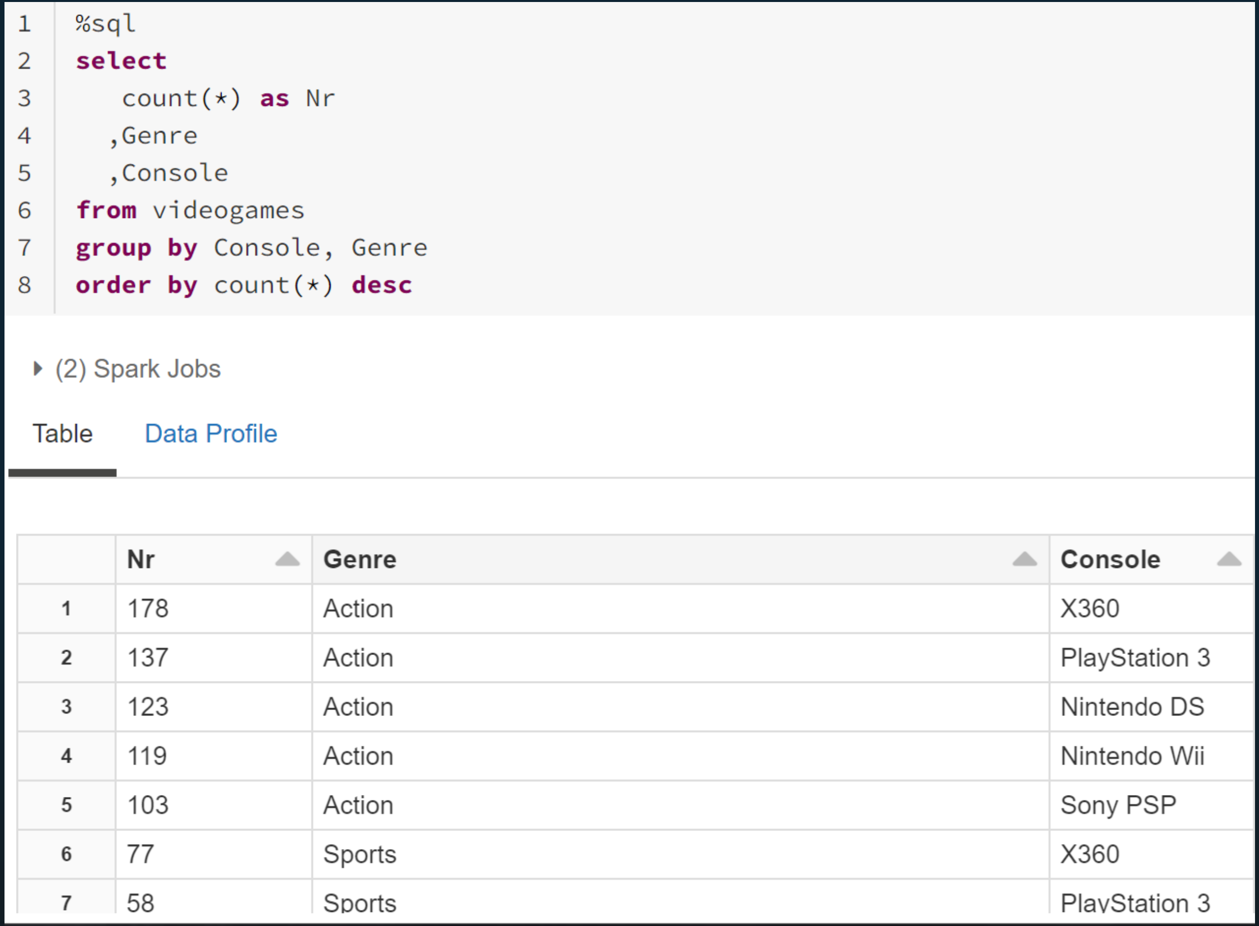
Spark + Databricks





```
spark.read.format("csv") \
  .option("header", "true") \
  .option("inferSchema", "true") \
  .load("/mnt/lake/bits/videogames.csv") \
  .display()
```

	Console	Title
1	Nintendo DS	Super Mario 64 DS
2	Sony PSP	Lumines: Puzzle Fusion

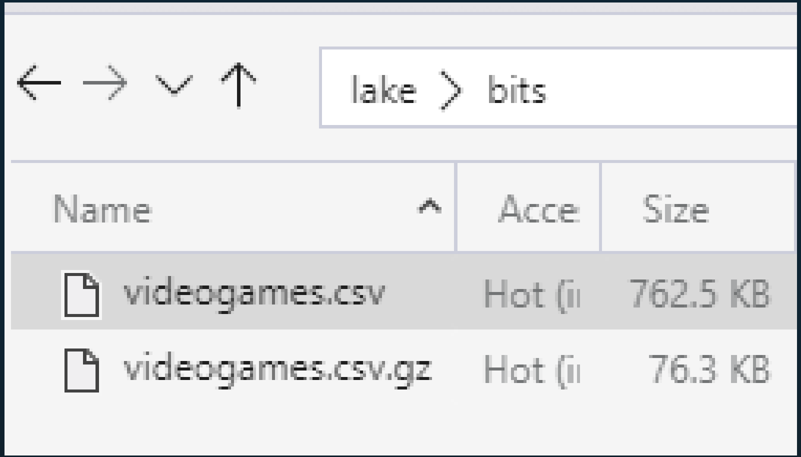


```
%sql
select
    count(*) as Nr
   ,Genre
   ,Console
from videogames
group by Console, Genre
order by count(*) desc
```

(2) Spark Jobs

Table   Data Profile

	Nr	Genre	Console
1	178	Action	X360
2	137	Action	PlayStation 3
3	123	Action	Nintendo DS
4	119	Action	Nintendo Wii
5	103	Action	Sony PSP
6	77	Sports	X360
7	58	Sports	PlayStation 3



lake > bits

Name	Acce	Size
videogames.csv	Hot (i	762.5 KB
videogames.csv.gz	Hot (i	76.3 KB

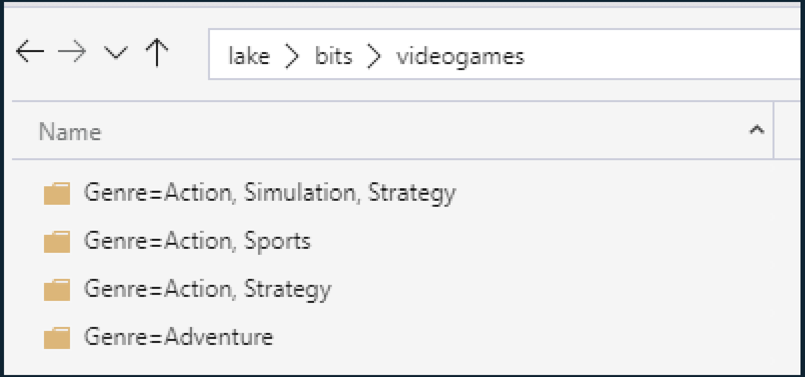
Possible Issues

- No schema (only column names)
- Have to uncompress on one node
- Have to read everything every time
- Only flat data, no nesting possible

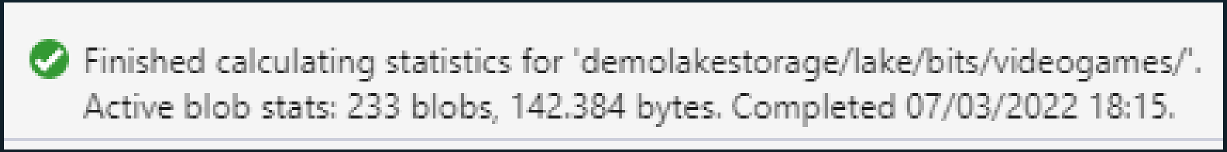
Possible Issues

- No schema (only column names)
- Have to uncompress on one node
- **Have to read everything every time**
- Only flat data, no nesting possible

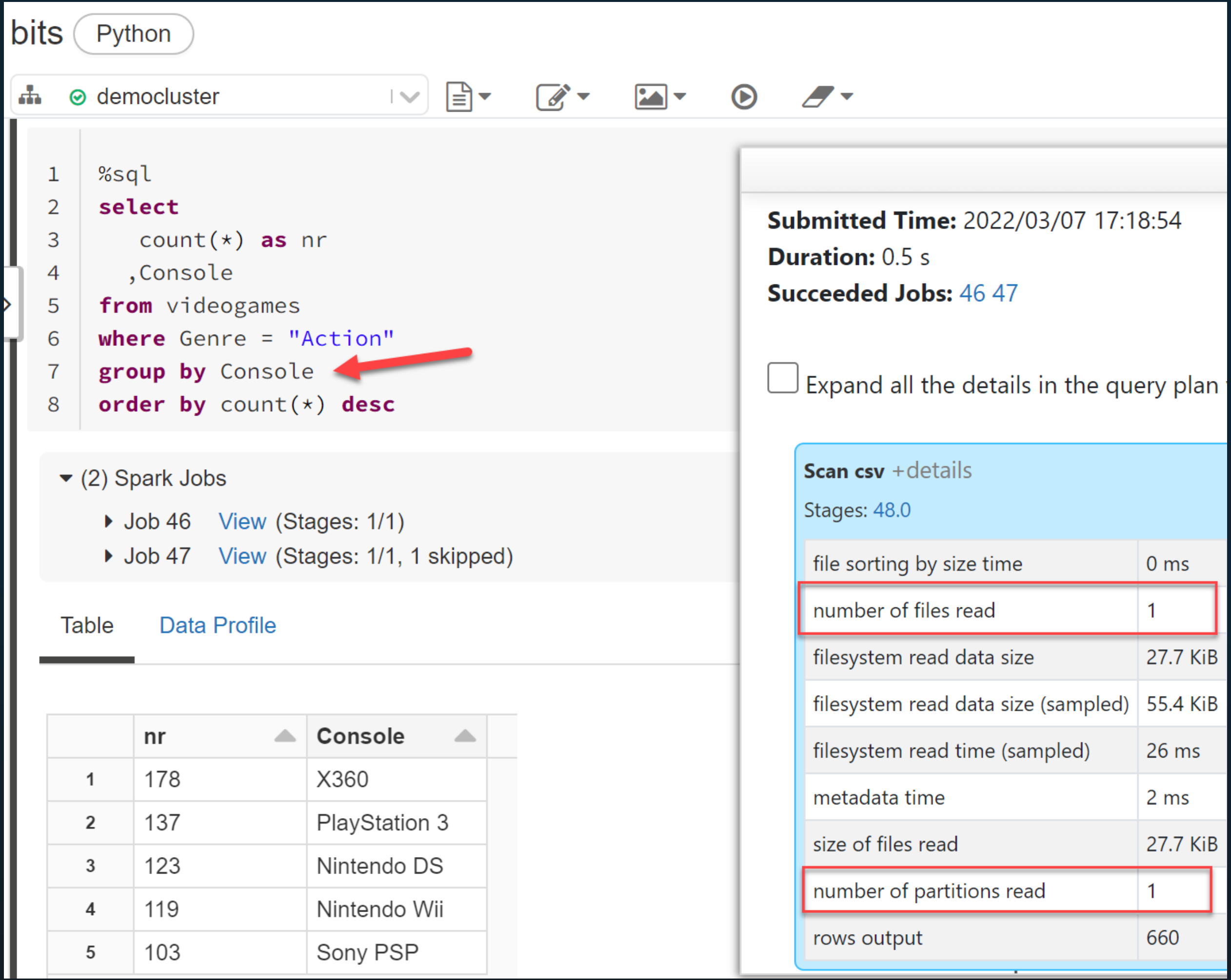
.partitionBy("Genre")



Genre=Action, Simulation, Strategy
Genre=Action, Sports
Genre=Action, Strategy
Genre=Adventure



Finished calculating statistics for 'demolakestorage/lake/bits/videogames/'. Active blob stats: 233 blobs, 142.384 bytes. Completed 07/03/2022 18:15.



bits Python

democluster

```
%sql
select
   count(*) as nr
  ,Console
from videogames
where Genre = "Action"
group by Console
order by count(*) desc
```

(2) Spark Jobs

- Job 46 View (Stages: 1/1)
- Job 47 View (Stages: 1/1, 1 skipped)

Table | Data Profile

	nr	Console
1	178	X360
2	137	PlayStation 3
3	123	Nintendo DS
4	119	Nintendo Wii
5	103	Sony PSP

Submitted Time: 2022/03/07 17:18:54
Duration: 0.5 s
Succeeded Jobs: 46 47

Expand all the details in the query plan

Scan csv +details

Stages: 48.0

file sorting by size time	0 ms
number of files read	1
filesystem read data size	27.7 KiB
filesystem read data size (sampled)	55.4 KiB
filesystem read time (sampled)	26 ms
metadata time	2 ms
size of files read	27.7 KiB
number of partitions read	1
rows output	660

Possible Issues

- No schema (only column names)
- Have to uncompress on one node
- Have to read everything every time
- **Only flat data, no nesting possible**

JSON

Columns instead of Rows

	Col A	Col B	Col C	Col D
Row 0	A0	B0	C0	D0
Row 1	A1	B1	C1	D1
Row 2	A2	B2	C2	D2
Row 3	A3	B3	C3	D3
Row 4	A4	B4	C4	D4

	Col A	Col B	Col C	Col D
Row 0	A0	B0	C0	D0
Row 1	A1	B1	C1	D1
Row 2	A2	B2	C2	D2
Row 3	A3	B3	C3	D3
Row 4	A4	B4	C4	D4

A0	A1	A2	A3	A4	B1	B2	B3
B4	C0	C1	C2	C3	C4	D0	D1
D2	D3	D4					

	Col A	Col B	Col C	Col D
Row 0	A0	B0	C0	D0
Row 1	A1	B1	C1	D1
Row 2	A2	B2	C2	D2
Row 3	A3	B3	C3	D3
Row 4	A4	B4	C4	D4

Apache Parquet

Born out of a cooperation between Twitter & Cloudera
Open source storage format: PAX – Partition Attributes Across

Data Page Layouts for Relational Databases
on Deep Memory Hierarchies
Anastassia Ailamaki
David J. DeWitt
Mark D. Hill
The VLDB Journal 11 (2002)

PAR1

-

-

-

-

-

PAR1

PAR1

-

-

-

-

Footer Length (4 bytes)

PAR1

PAR1

-

-

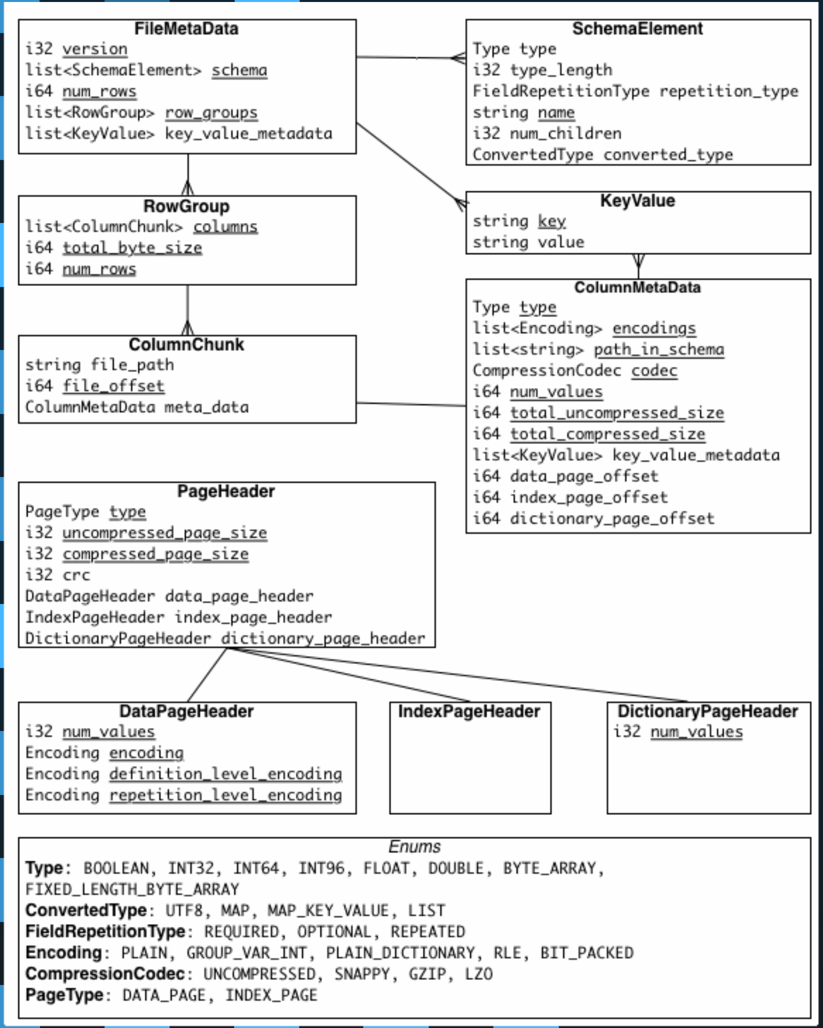
-

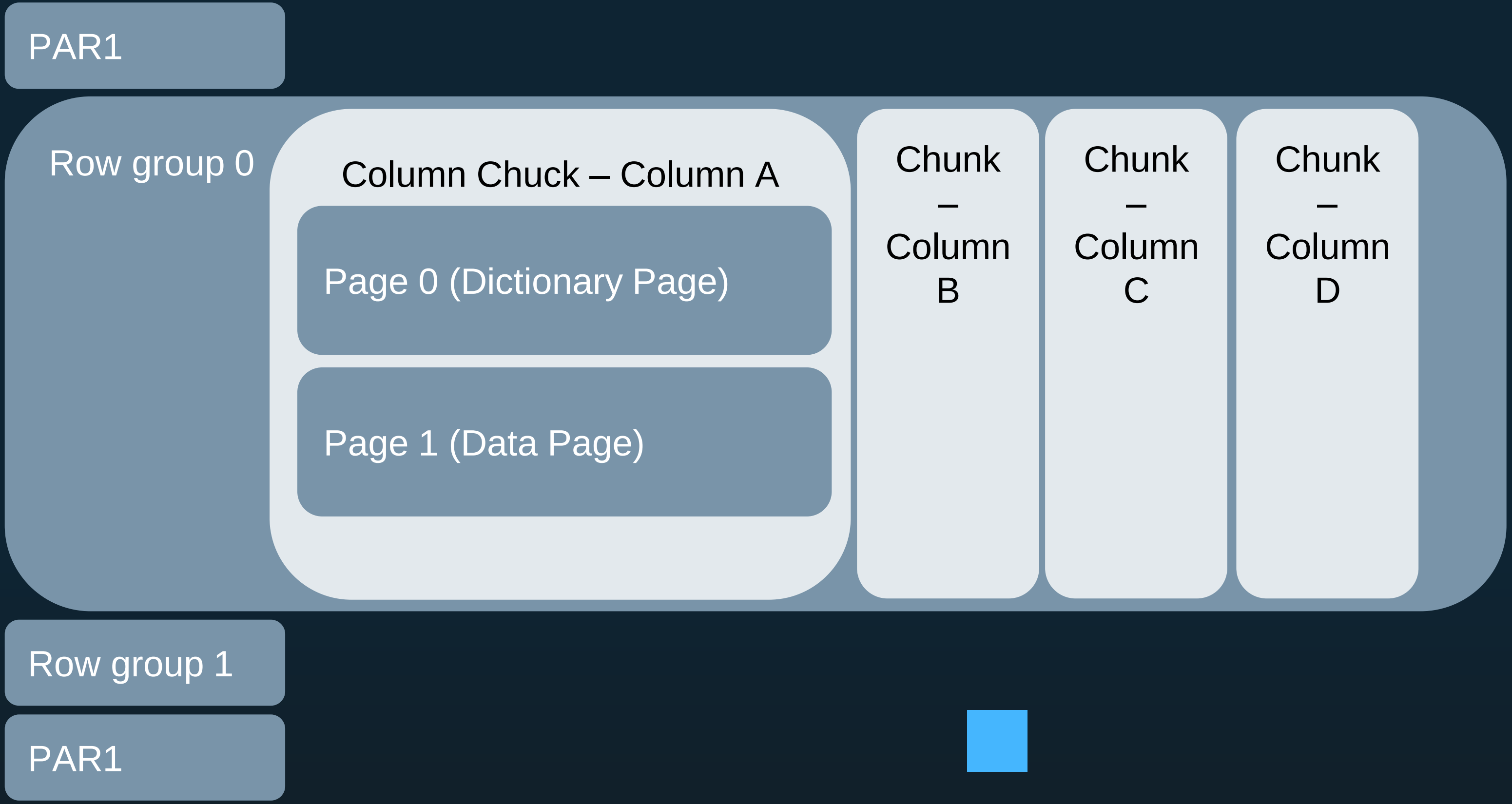
Footer (metadata)

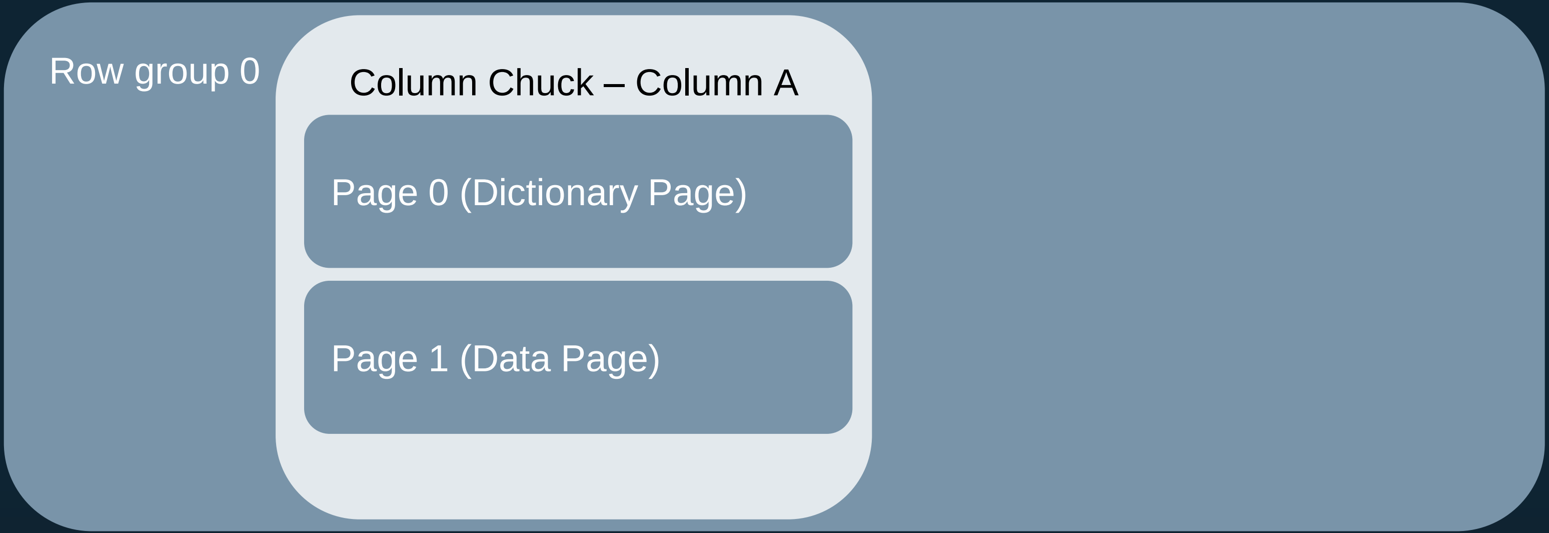
Footer Length (4 bytes)

PAR1

Thrift Compact Protocol


FileMetaData
i32 version
list<SchemaElement> schema
i64 num_rows
list<RowGroup> row_groups
list<KeyValue> key_value_metadata
SchemaElement
Type type
i32 type_length
FieldRepetitionType repetition_type
string name
i32 num_children
ConvertedType converted_type
RowGroup
list<ColumnChunk> columns
i64 total_byte_size
i64 num_rows
KeyValue
string key
string value
ColumnChunk
string file_path
i64 file_offset
ColumnMetaData meta_data
ColumnMetaData
Type type
list<Encoding> encodings
list<string> path_in_schema
CompressionCodec codec
i64 num_values
i64 total_uncompressed_size
i64 total_compressed_size
list<KeyValue> key_value_metadata
i64 data_page_offset
i64 index_page_offset
i64 dictionary_page_offset
PageHeader
PageType type
i32 uncompressed_page_size
i32 compressed_page_size
i32 crc
DataPageHeader data_page_header
IndexPageHeader index_page_header
DictionaryPageHeader dictionary_page_header
DataPageHeader
i32 num_values
Encoding encoding
Encoding definition_level_encoding
Encoding repetition_level_encoding
IndexPageHeader
DictionaryPageHeader
i32 num_values
Enums
Type: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BYTE_ARRAY, FIXED_LENGTH_BYTE_ARRAY
ConvertedType: UTF8, MAP, MAP_KEY_VALUE, LIST
FieldRepetitionType: REQUIRED, OPTIONAL, REPEATED
Encoding: PLAIN, GROUP_VAR_INT, PLAIN_DICTIONARY, RLE, BIT_PACKED
CompressionCodec: UNCOMPRESSED, SNAPPY, GZIP, LZO
PageType: DATA_PAGE, INDEX_PAGE


PAR1
Row group 0
Column Chuck – Column A
Page 0 (Dictionary Page)
Page 1 (Data Page)
Chunk – Column B
Chunk – Column C
Chunk – Column D
Row group 1
PAR1


Row group 0
Column Chuck – Column A
Page 0 (Dictionary Page)
Page 1 (Data Page)

Page 1

Metadata

Min / Max / Count

Encoding and Compression Codec info

Repetition levels

(only used for nested columns)

Definition levels

(only used for nullable columns)

Encoded Data

3 Types of Metadata

- File metadata
- Column (chunk) metadata
- Page header metadata

2 Locations for metadata

Page Header

File Footer

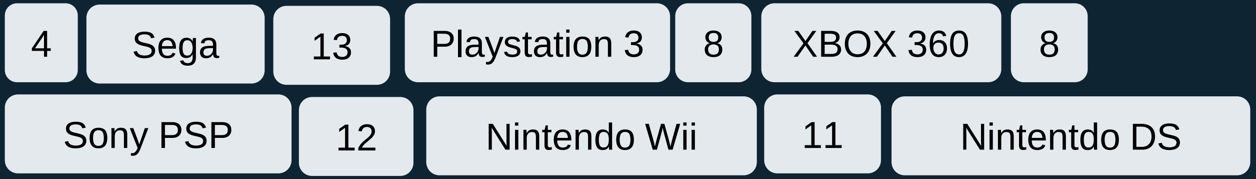
Advantages So Far:

- Metadata at the end, one pass write.
- Column locations known, can grab only the columns needed.
- Min / Max values known, can skip pages, or even whole files.
- (Parquet table is usually a bunch of files and not just one)
- Directory based partitioning used to skip files, don’t even have to parse the metadata.
- Data portion of pages are compressed, allowing compression without having to resort to a single node processor.

Encoding

Plain

- Fixed-width: back-to-back
- Variable Length: prefixed



RLE_Dictionary

- Run-length encoding
- Bit packing
- Dictionary Compression

Sony PSP
XBOX 360
SEGA
Nintendo Wii
Nintendo Wii
Nintendo Wii
Nintendo DS
Nintendo DS

0	Sony PSP
1	XBOX 360
2	SEGA
3	Nintendo Wii
4	Nintendo DS

0	1	2	3	3
3	4	4		

0	Sony PSP
1	XBOX 360
2	SEGA
3	Nintendo Wii
4	Nintendo DS

0	1	2	3,3	2,4

Delta_Length_Byte_Array

V2

4,13,8,8,12,11	SegaPlaystation 3XBOX 360Sony PSPNintendo WiiNintendo DS

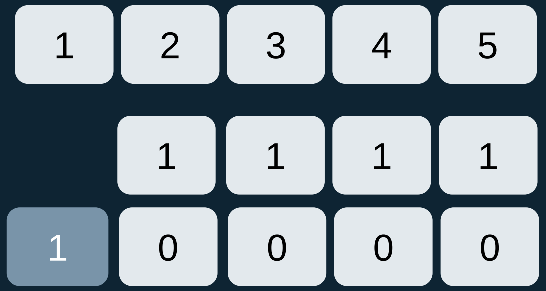
V2

Delta_Binary_Packed

7	5	3	1	1	2	3	4
	-2	-2	-2	1	1	1	1
-2	0	0	0	3	3	3	3

Value Count : 8
First Value : 7
Minimum delta: -2
Bitwidth : 00 00 00 11 11 11

Delta_Binary_Packed



Value Count : 5
First Value : 1
Minimum delta: 1
Bitwidth : Not Needed!!

Adapted from “Decoding billions of integers per second through vectorization (D. Lemire, L.Boytsov)

Random Learnings

Dictionary pages are always PLAIN encoded
Delta Encoding with signed integers is called “ZigZag Encoding”
Row-Groups are 128MB, pages are 1MB by default (In spark I guess)

Is V2 The Future?

Maybe, a bit like IPv6 perhaps?
Check out Databricks Delta though!