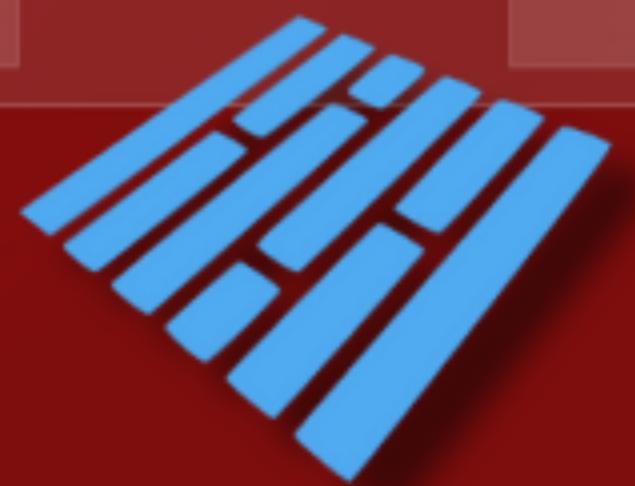




INTRODUCTION TO Parquet



AMR ELHELW

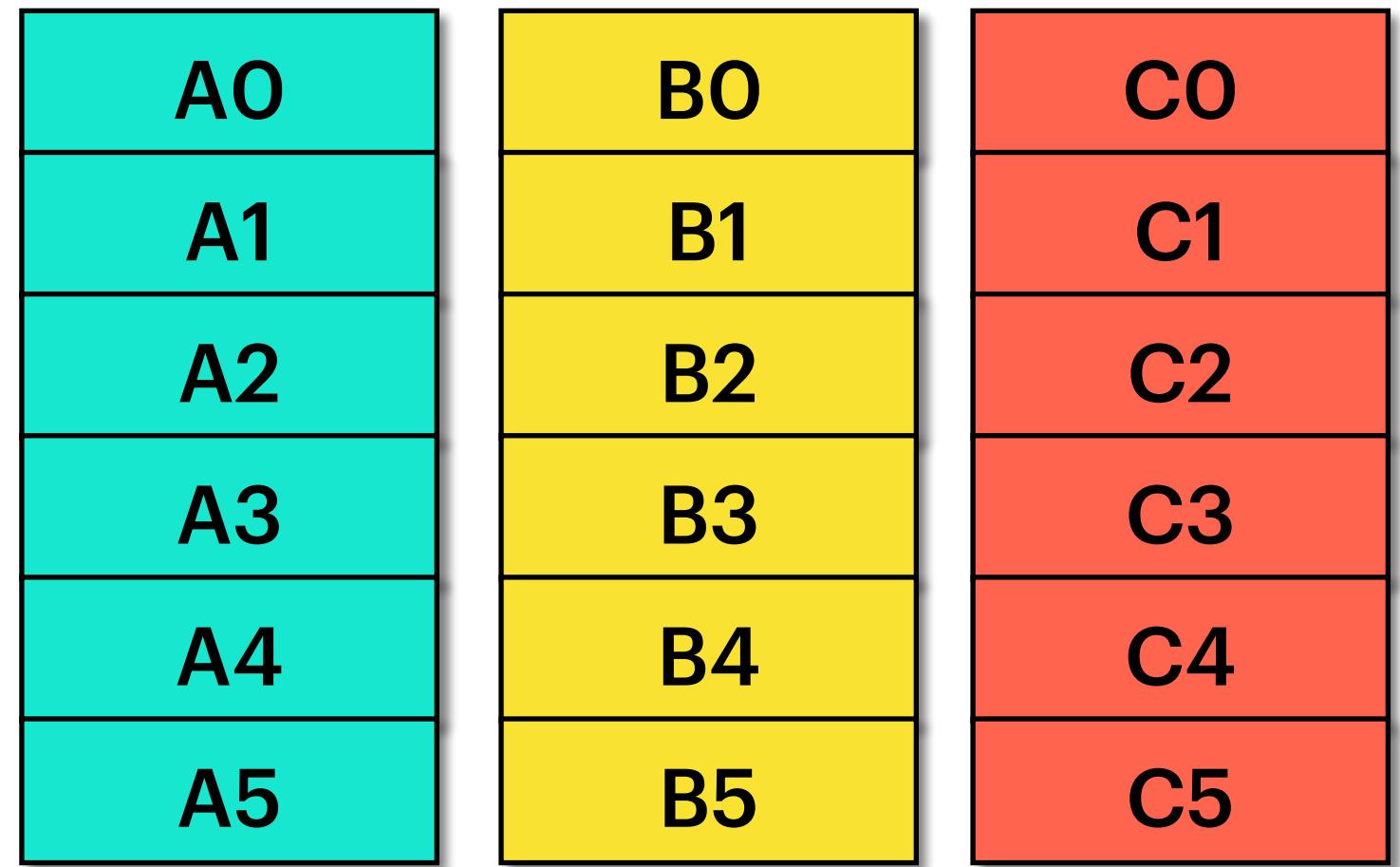
	Col A	Col B	Col C
Row 0	A0	B0	C0
Row 1	A1	B1	C1
Row 2	A2	B2	C2
Row 3	A3	B3	C3
Row 4	A4	B4	C4
Row 5	A5	B5	C5

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Row-oriented

Physical Storage

A0	B0	C0	A1	B1	C1	A2	B2	C2	A3	B3	C3	A4	B4	C4	A5	B5	C5
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



Column-oriented (Columnar)

Physical Storage

A0	A1	A2	A3	A4	A5	B0	B1	B2	B3	B4	B5	C0	C1	C2	C3	C4	C5
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

OLTP

- Access a *small number of rows*
- *Read/Update several attributes from the same row.*
- *Primary focus is operational*

```
select *  
from employees  
where dept = 'IT'
```

OLAP

- *Read a lot of rows (usually very large tables)*
- *Focus on a small number of columns in order to do some kind of aggregation*
- *Primary focus here is analytics*

```
select avg(salary)  
from employees
```

```
select *  
from employees  
where dept = 'IT'
```

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Row-oriented

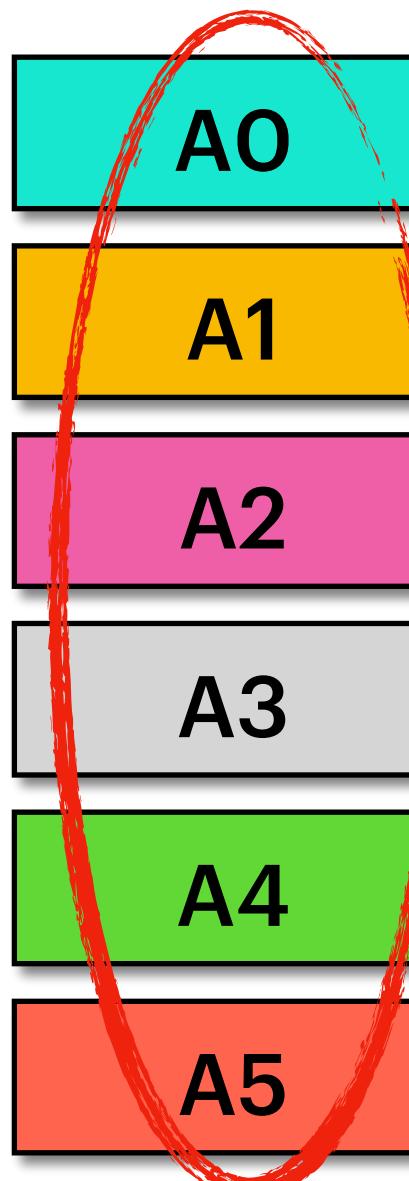
OLTP

A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Columnar

OLTP

```
select avg(salary)  
from employees
```



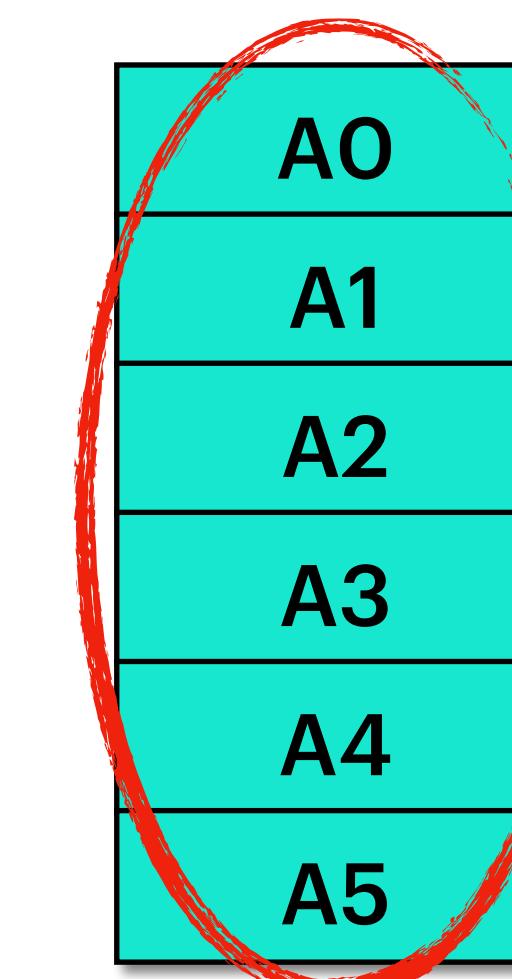
A0	B0	C0
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

Row-oriented

OLTP



OLAP



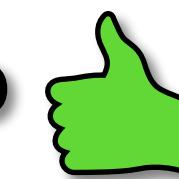
A0	BO	CO
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4
A5	B5	C5

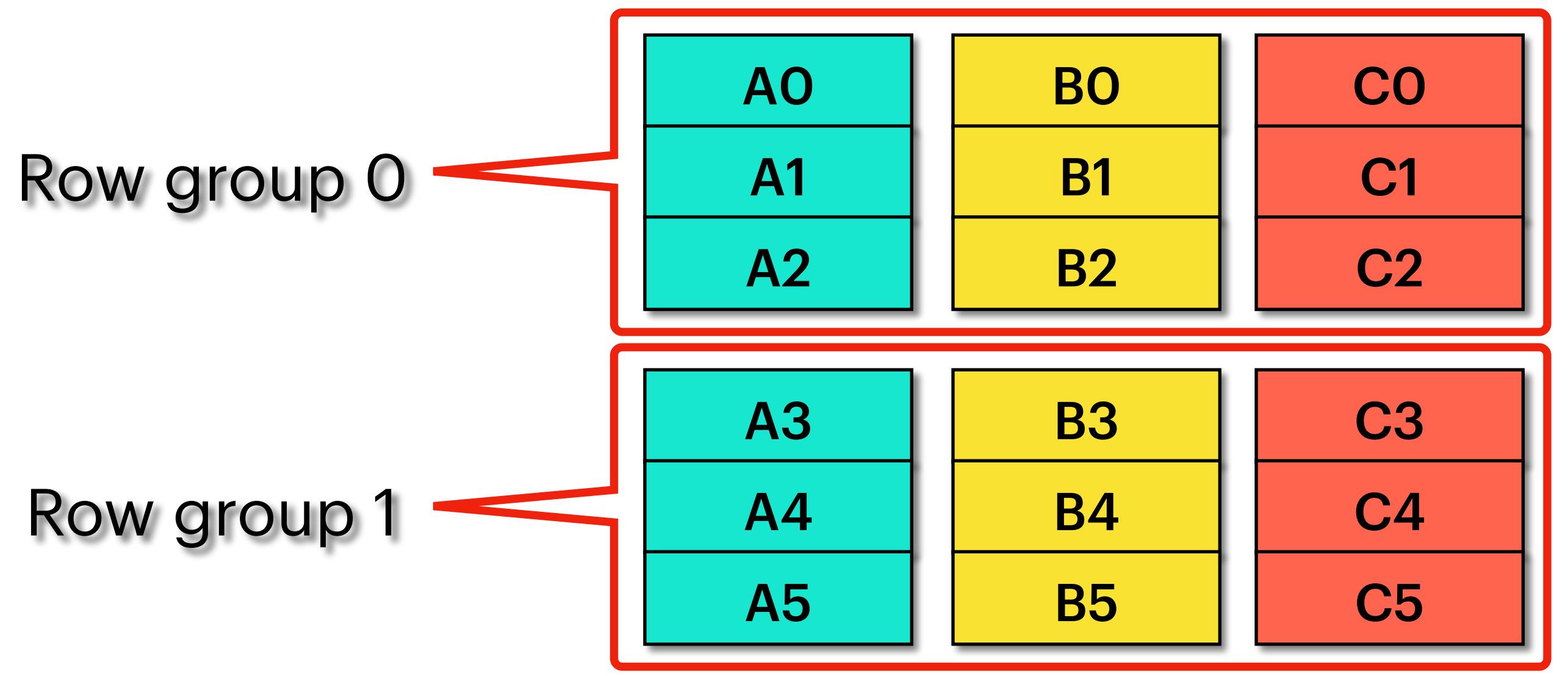
Columnar

OLTP



OLAP





Physical Storage

A0	A1	A2	B0	B1	B2	C0	C1	C2	A3	A4	A5	B3	B4	B5	C3	C4	C5
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



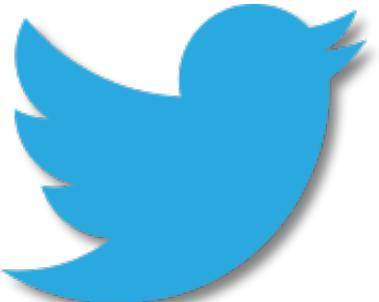
<https://parquet.apache.org/docs/overview>

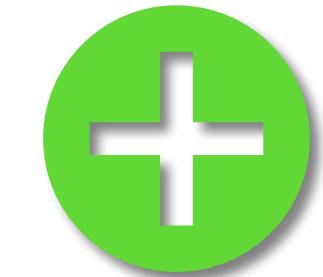
[Documentation](#) / Overview

Overview

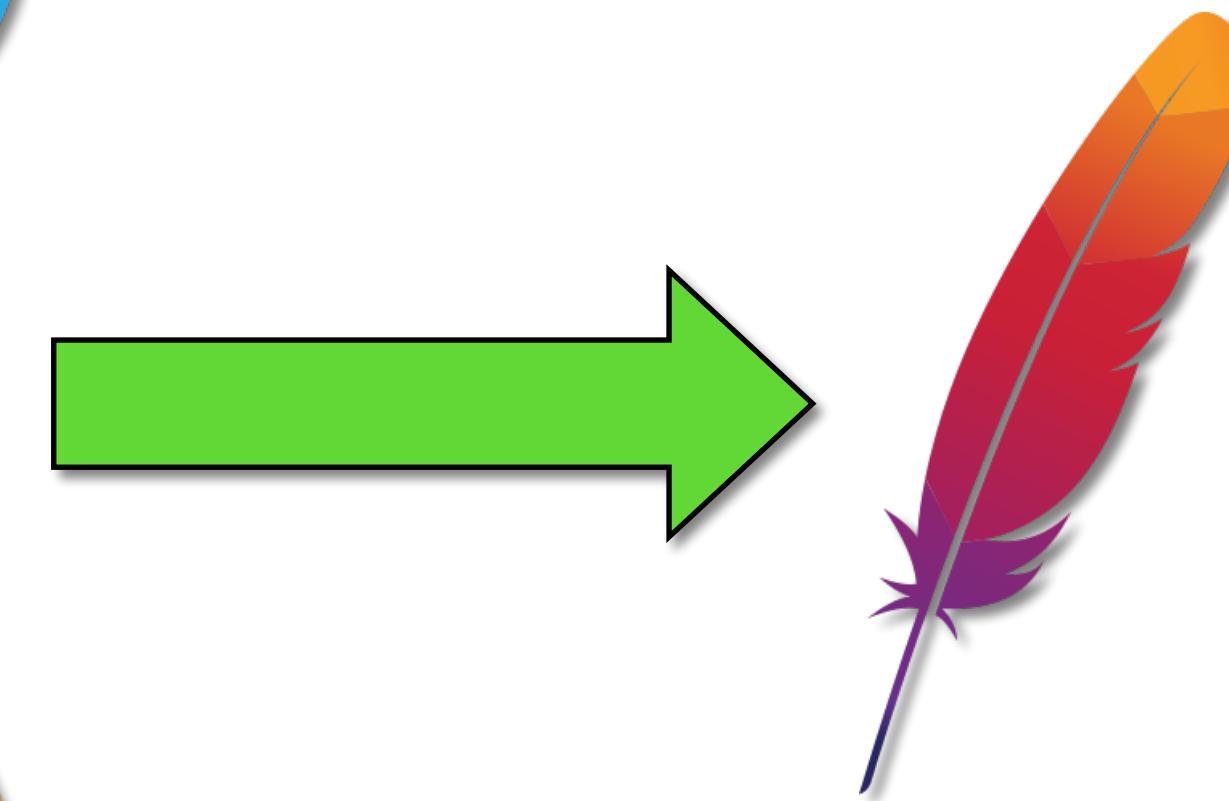
All about Parquet.

Apache Parquet is an **open source**, **column-oriented data file format** designed for **efficient data storage and retrieval**. It provides high performance **compression and encoding** schemes to handle **complex data** in bulk and is supported in **many programming language and analytics tools**.

twitter 



CLOUDERA



THE
APACHE[®]
SOFTWARE FOUNDATION

Parquet File

Row Group

Column
Chunk

Column
Chunk

Column
Chunk

...

Column
Chunk

Row Group

Column
Chunk

Column
Chunk

Column
Chunk

...

Column
Chunk

⋮

Row Group

Column
Chunk

Column
Chunk

Column
Chunk

...

Column
Chunk

Footer

Column Chunk

Page

Page

Page

Page

Page

Page



Footer Contents

- Schema (Column names, types, etc.)
- Total number of rows in the file
- **Row group metadata (for each Row Group)**
 - # rows, byte size, file offset
 - **Column Chunk metadata (for each Column Chunk)**
 - File offset, chunk size
 - # distinct values in chunk, # nulls in chunk
 - Min/max values in chunk
 - Encoding/compression info for each page

Parquet File - Table: Employee

Row Group 0

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 1

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 2

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 3

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Footer

```
select avg(salary)  
from employees
```

Parquet File - Table: Employee

Row Group 0

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 1

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 2

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 3

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Footer

```
select avg(salary)  
from employee  
where id between 400 and 600
```

Row group 0:

"id": min = 102, max = 256

"dept": min = 10, max = 55

...

Row group 1:

"id": min = 260, max = 529

"dept": min = 5, max = 48

...

Row group 2:

"id": min = 570, max = 863

"dept": min = 18, max = 57

...

Row group 3:

"id": min = 869, max = 1104

"dept": min = 12, max = 41

...



Parquet File - Table: Employee

Row Group 0

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 1

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 2

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Row Group 3

Column
Chunk
"id"

Column
Chunk
"Name"

Column
Chunk
"Dept"

Column
Chunk
"Salary"

Footer

```
select avg(salary)  
from employee  
where dept = 30
```

Row group 0:

"id": min = 102, max = 256

"dept": min = 10, max = 55

...

Row group 1:

"id": min = 260, max = 529

"dept": min = 5, max = 48

...

Row group 2:

"id": min = 570, max = 863

"dept": min = 18, max = 57

✓

...

Row group 3:

"id": min = 869, max = 1104

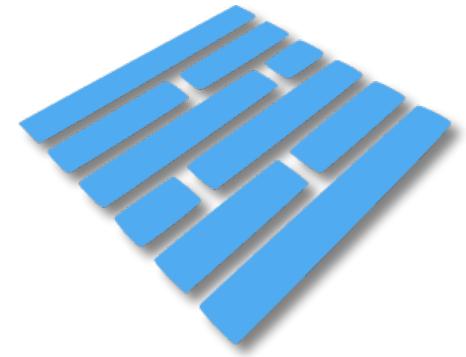
"dept": min = 12, max = 41

✓

...

Note on file size

- **Many small files**
 - Overhead of fetching multiple files
 - Setting up reader objects
 - Too many footers to read
- **Few huge files**
 - Footer gets too large — too many row groups, with info for each
- **No “magic number” for how big a file should be**
 - Trial and error
 - System specific guidelines

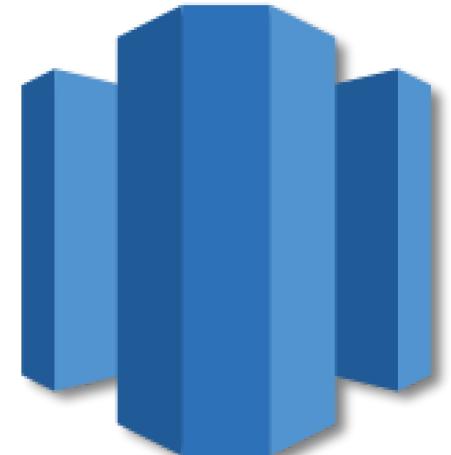


Parquet

SQL



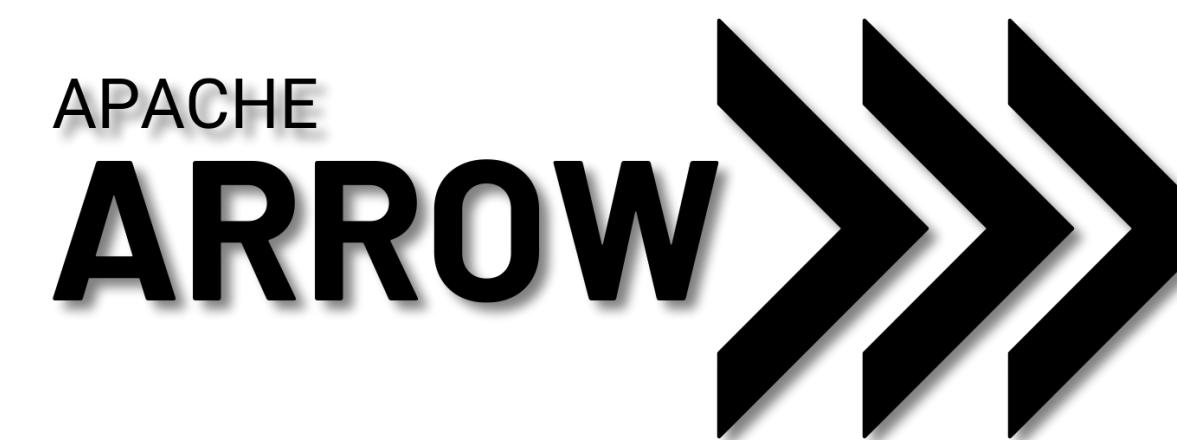
Google
Big Query



amazon
REDSHIFT



Programming Languages



C, C++, C#, Go,
Java, JavaScript,
Python, R, Ruby,
Rust ...

Python

Limitations

- Overhead for small datasets
 - Metadata adds cost
- Not ideal for frequent appends/updates
 - Requires rewrite
- Not suited for OLTP workloads
 - Inefficient for Row-Level Operations