NYU | Center for Data Science

Week 06.2: Dremel and Parquet

DS-GA 1004: Big Data

Today’s plan

- Background on column stores
- **Dremel and Parquet**

TLDR: parallelism isn’t everything.

Data structures are still important!

Dremel

[Melnik et al., 2010]

- Low-latency query system for read-only, **structured data**
- Developed at Google ~2006-2010
- Lots of cool ideas in the paper, but we'll focus on the **data format**
- Core ideas were quickly adopted and re-implemented in **Parquet** (2013)
  - Parquet is the default storage format for Spark

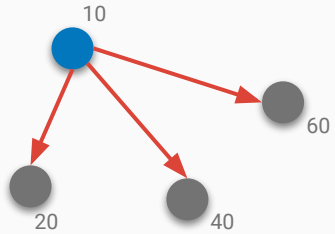
Nested and structured data

- Not everything fits nicely in relations
- Variable-length/depth can be difficult
- Record-oriented storage is more natural here

How can we get all the benefits of column stores but for structured data?

Example: web documents

- DocID [required]
- Links [optional]
  - Backward [0 or more]
  - Forward [0 or more]
- Name [1 or more]
  - Language [1 or more]
    - Code [required]
    - Country [optional]
  - URL [optional]




```
DocID: 10
Links:
      Forward: 20
      Forward: 40
      Forward: 60
Name:
      Language:
            Code: ‘en-us’
            Country: ‘us’
      Language:
            Code: ‘en’
      URL: ‘http://A’
Name:
      URL: ‘http://B’
Name:
      Language:
            Code: ‘en-gb’
            Country: ‘gb’
```

Example: web documents

- DocID [required]
- Links [optional]
  - Backward [0 or more]
  - Forward [0 or more]
- Name [1 or more]
  - Language [1 or more]
    - Code [required]
    - Country [optional]
  - URL [optional]

Field names are ***paths***, e.g.:

```
DocID
Links.Forward
Name.Language.Code
```

```
DocID: 10
Links:
      Forward: 20
      Forward: 40
      Forward: 60
Name:
      Language:
            Code: ‘en-us’
            Country: ‘us’
      Language:
            Code: ‘en’
      URL: ‘http://A’
Name:
      URL: ‘http://B’
Name:
      Language:
            Code: ‘en-gb’
            Country: ‘gb’
```

Record flattening

- **Key idea**: track repetitions of fields within a record
- ***Repetition level (r)***: which level repeated most recently?
- ***Definition level (d)***: how many optional fields in the path are present?
- **Required fields ⇒ Same levels as parent**
- **Optional fields ⇒ Same r-level as parent, d-level increments**
- **Repeated fields ⇒ r-level and d-level both increment from parent**

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

Flattening example

Node.DocID

value	r	d

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d

Node.Links.Backward

value	r	d

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d

Node.Links.Backward

value	r	d

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

DocID is required

r=0, d=0

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2

Node.Links.Backward

value	r	d

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

Links is **optional** (but present)

First occurrence ⇒ r=0

Links.Forward is a **repeated** field
Forward ⇒ d=2

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2

Node.Links.Backward

value	r	d

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

...Forward ⇒ d=2

Repetition in level r=1

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

...Forward ⇒ d=2

Repetition in level r=1

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d

Node.Name.Language.Country

value	r	d

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

Links.Backward is **repeated** (but absent)

r=0, d=1

No value in this record, so fill a NULL

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

Node.Name.Language.Code

value	r	d
en-us	0	2

Node.Name.Language.Country

value	r	d

Name.Language.Code required

First occurrence (r=0)
Full definition path (d=2)

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2

Node.Name.Language.Country

value	r	d
us	0	3

...Country is optional ⇒ d=3

First occurrence (r=0)
Full definition path (d=3)

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2

Node.Name.Language.Country

value	r	d
us	0	3

...Code is required

Repetition at r=2 (Name.Language)

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2

...Language.Country optional

Repeated at Language level
r=2, d=2

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: 'en-us'
                Country: 'us'
        Language:
                Code: 'en'
        URL: 'http://A'
Name:
        URL: 'http://B'
Name:
        Language:
                Code: 'en-gb'
                Country: 'gb'
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: 'http://C'
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: 'en-us'
                Country: 'us'
        Language:
                Code: 'en'
        URL: 'http://A'
Name:
        URL: 'http://B'
Name:
        Language:
                Code: 'en-gb'
                Country: 'gb'
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: 'http://C'
```

Node.Name.URL is optional
⇒ d=2

No repetitions: r=0

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2

Node.Links.Forward

value	r	d
20	0	2
40		
60		

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name ⇒ d=1

But no Language.* data...

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: 'en-us'
                Country: 'us'
        Language:
                Code: 'en'
        URL: 'http://A'
Name:
        URL: 'http://B'
Name:
        Language:
                Code: 'en-gb'
                Country: 'gb'
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: 'http://C'
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NU		

Node.Name.URL ⇒ d=2

Repetition at r=1 (Node.Name)

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NU		

...Language.Code ⇒ d=2

Repetition at r=1 (Node.Name)

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: 'en-us'
                Country: 'us'
        Language:
                Code: 'en'
        URL: 'http://A'
Name:
        URL: 'http://B'
Name:
        Language:
                Code: 'en-gb'
                Country: 'gb'
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: 'http://C'
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2

Node.Links.Forward

value	r	d
20	0	2
40		
60		

Node.Links.Backward

value	r	d
NULL	0	1

...Language.Country ⇒ d=3

Repetition at r=1 (Node.Name)

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: 'en-us'
                Country: 'us'
        Language:
                Code: 'en'
        URL: 'http://A'
Name:
        URL: 'http://B'
Name:
        Language:
                Code: 'en-gb'
                Country: 'gb'
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: 'http://C'
```

Flattening example

Node.DocID

value	r	d
10	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NU		

Node.Name ⇒ d=1

No URL data

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0
20	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NU		

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

Node.DocID ⇒ d=0

Required field, new document (r=0)

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0
20	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2

Node.Links.Backward

value	r	d
NULL	0	1
10	0	2

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NU		

Node.Links.Backward ⇒ d=2

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Flattening example

Node.DocID

value	r	d
10	0	0
20	0	0

Node.Name.URL

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Node.Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Node.Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

Node.Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Node.Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
Name:
        Language:
                Code: ‘en-us’
                Country: ‘us’
        Language:
                Code: ‘en’
        URL: ‘http://A’
Name:
        URL: ‘http://B’
Name:
        Language:
                Code: ‘en-gb’
                Country: ‘gb’
```

… and all the rest

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
Name:
        URL: ‘http://C’
```

Partial record assembly

- Dremel can rebuild **partial views** (projections) of the data easily
- Unused attributes can be ignored!
- But decoding is **inherently sequential** ⇒ difficult to parallelize

Node.DocID

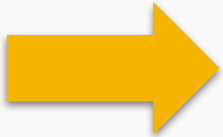
value	r	d
10	0	1
20	0	1

Node.Links.Forward

value	r	d
20	0	2
40	2	2
60	2	2
80	0	2

Node.Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2



```
DocID: 10
Links:
        Forward: 20
        Forward: 40
        Forward: 60
```

```
DocID: 20
Links:
        Backward: 10
        Backward: 30
        Forward:  80
```

After flattening...

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

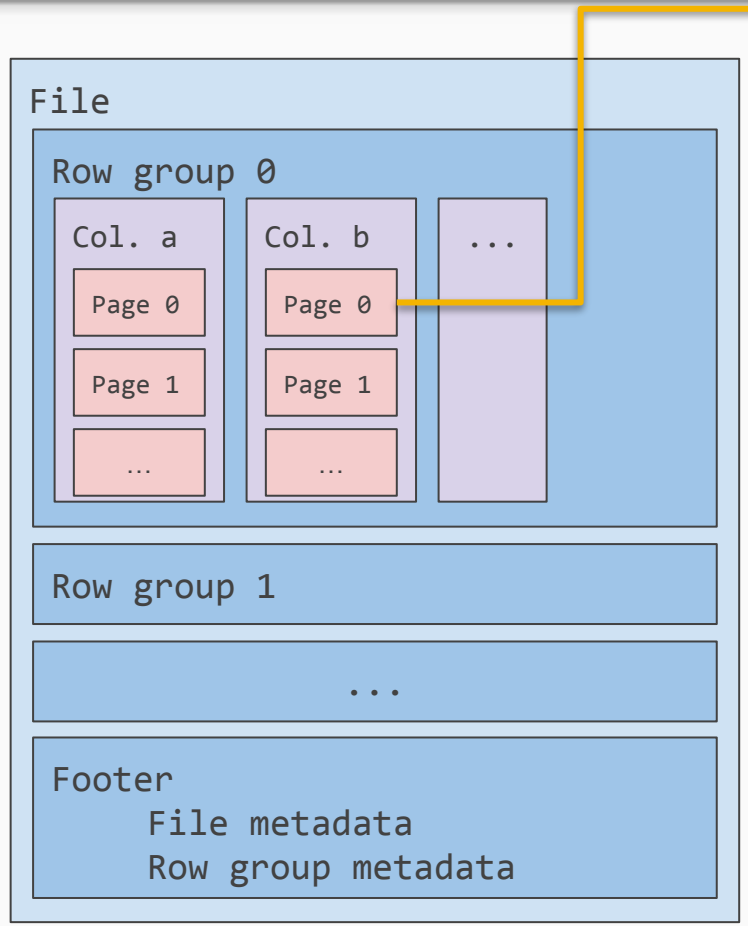
- Repetition and definition columns are highly compressible
  - Not even needed for complete, tabular data!
- Value fields are now columnar
  - May also be compressed
- Columns are broken into **blocks** and compressed independently
  - This alleviates some decoding complexity and improves parallelism

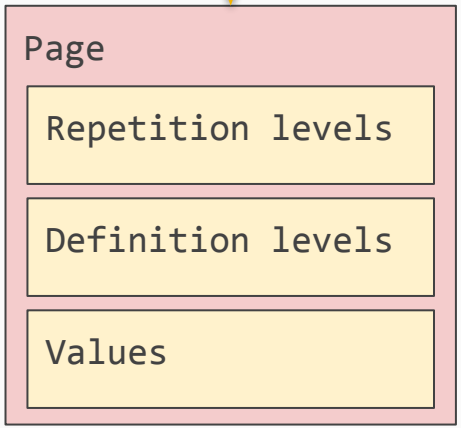
Parquet

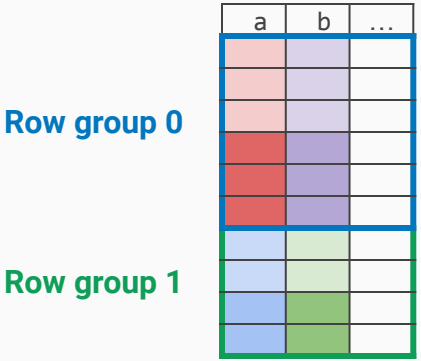
Parquet

- Developed at Twitter and Cloudera, v1.0 in 2013
- Now an Apache project, and the default/recommended storage for Spark
- Based on Dremel flattening, but without the analysis engine
- Name comes from the shape of the data:
  - blocks of column fragments

Parquet format







- Pages for a column are compressed independently
- Small pages make it easier to read single records, but incur more overhead
- Row groups should be large, but fit in one HDFS block

https://parquet.apache.org/documentation/latest/

Cool things about Parquet

- Cross-platform, cross-language support
  - Java, C++, Python, Scala, …
- Allows **partial decoding** (only decode necessary columns)
- Integrates nicely with Spark and HDFS
  - Preserves RDD / DataFrame schema directly
  - HDFS block-aware layout
  - Partition discovery / exposes control over partitions by column

Using parquet in practice (with Spark)

- Column efficiency depends on row ordering
- DataFrame partitions can be written out separately
    - Remember: partitions are similar to RDBMS indices; they can help locate records!

Other column formats / implementations

- Most DataFrame implementations are columnar (pandas, R)
  - This is the most reasonable way to handle mixed-type data!
- **Apache Arrow** is a unified API for in-memory column stores
  - Makes it easier to exchange data between Spark / Pandas / Rapids / etc

Wrap-up

- Column stores can improve speed for attribute (rather than record)-oriented analyses
- Dremel turns structured or variable-length data into columnar representations
- Parquet provides an open source reimplementation of the Dremel format