EFFICIENT COLUMNAR STORAGE
WITH
APACHE PARQUET

Ranganathan Balashanmugam, Aconex

Apache: Big Data North America 2017
“The Tables Have Turned.”
“The Tables Have Turned.” - 90°
### Simple Structure

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
</tbody>
</table>

**Row:**

- `a1`, `b1`, `c1`
- `a2`, `b2`, `c2`
- `a3`, `b3`, `c3`

**Columnar:**

- `a1`, `a2`, `a3`, `b1`, `b2`, `b3`, `c1`, `c2`, `c3`
“Optimizing the disk seeks.”
“The Tables Have Turned.” - 90°
SIMPLE STRUCTURE

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
</tbody>
</table>

row

| a1 | b1  | c1  | a2 | b2  | c2  | a3 | b3  | c3  |

columnar

| a1 | a2  | a3  | b1 | b2  | b3  | c1 | c2  | c3  |
NESTED AND REPEATED STRUCTURES

How to preserve in column store?

```protobuf
textproto
docid: 10
links:
  forward: 20
  forward: 40
  forward: 60
name:
  language:
    code: 'en-us'
    country: 'us'
name:
  language:
    code: 'en'
    url: 'http://a'
name:
  language:
    code: 'en-gb'
    country: 'gb'
```
“Get these performance benefits for nested structures into Hadoop ecosystem.”
MOTIVATION

- Allow complex nested data structures
- Very efficient compression and encoding schemes
- Support many frameworks
“Columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.”
DESIGN GOALS

- Interoperability
- Space efficiency
- Query efficiency
object models

- avro
- thrift
- protobuf
- pig
- hive
- scalding
- ...

object model converters

- parquet

storage format

- column readers

- parquet binary file

parquet file

header - Magic number (4 bytes) : “PAR1”

row group 0

- column a
  - Page 1
  - Page 2
  - … Page n

- column b
  - Page 1
  - Page 2
  - … Page n

- column c
  - Page 1
  - Page 2
  - … Page n

row group 1

…row group n

footer

* group of rows
* max size buffered while writing
* 50 MB - 1GB

* data of one column in row group
* can be read independently

* good enough for compression efficiency
* 8KB - 1 MB
* good enough to read

src: https://github.com/Parquet/parquet-format
### file metadata (ThriftCompactProtocol)

- version
- schema

#### row group 0 metadata
- total byte size
- total rows

<table>
<thead>
<tr>
<th>column 0</th>
<th>column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>type/path/encodings/codec</td>
<td></td>
</tr>
<tr>
<td>num of values</td>
<td></td>
</tr>
<tr>
<td>compressed/uncompressed size</td>
<td></td>
</tr>
<tr>
<td>data page offset</td>
<td></td>
</tr>
</tbody>
</table>

#### column 0
- ...                              | ...                               |

#### column 1
- ...                              | ...                               |

#### footer length (4 bytes)

#### Magic number (4 bytes): “PAR1”
encoded and compressed

**page header** (*ThriftCompactProtocol*)

- repetition levels
- definition levels
- values
```protobuf
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;
    }
    optional string Url;
  }
}
```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;
    }
    optional string Url;
  }
}

Document 1
DocId: 10
Links:
  Forward: 20
  Forward: 40
  Forward: 60
Name:
  Language:
    Code: 'en-us'
    Country: 'us'
  Language:
    Code: 'en'
Name:
  Url: 'http://a'

Document 2
DocId: 20
Links:
  Backward: 10
  Backward: 30
  Forward: 80
Name:
  Url: 'http://c'
“Can we represent it in columnar form efficiently and read them back to their original nested data structure?”
“Can we represent it in columnar former efficiently and read them back to their original nested data structure?”

Dremel encoding
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;
    }
    optional string Url;
  }
}

fill all the nulls
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Links.Forward</td>
<td>0</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Links.Forward[1]</td>
<td>1</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>Links.Forward[2]</td>
<td>1</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>Links.Forward</td>
<td>0</td>
<td>2</td>
<td>80</td>
</tr>
</tbody>
</table>
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

### COLUMNS

<table>
<thead>
<tr>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name.Language.Code</td>
<td>en-us</td>
<td></td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td>en</td>
<td></td>
</tr>
<tr>
<td>Name[1].Language[].Code</td>
<td>null</td>
<td></td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td>en-gb</td>
<td></td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td>null</td>
<td></td>
</tr>
</tbody>
</table>
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>2</td>
<td>en-us</td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td></td>
<td></td>
<td>en</td>
</tr>
<tr>
<td>Name[1].Language[1].Code</td>
<td></td>
<td></td>
<td>null</td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td></td>
<td></td>
<td>en-gb</td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td></td>
<td></td>
<td>null</td>
</tr>
</tbody>
</table>
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>2</td>
<td>en-us</td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td>2</td>
<td>2</td>
<td>en</td>
</tr>
<tr>
<td>Name[1].Language[1].Code</td>
<td>2</td>
<td>2</td>
<td>null</td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td>2</td>
<td>2</td>
<td>en-gb</td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td>2</td>
<td>2</td>
<td>null</td>
</tr>
</tbody>
</table>
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>2</td>
<td>en-us</td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td>2</td>
<td>2</td>
<td>en</td>
</tr>
<tr>
<td>Name[1].Language[1].[Code]</td>
<td>1</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td>1</td>
<td>1</td>
<td>en-gb</td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>0</td>
<td>null</td>
</tr>
</tbody>
</table>
In the path to the field, what is the last repeated field?

In the path to the field, how many defined fields?

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>2</td>
<td>en-us</td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td>2</td>
<td>2</td>
<td>en</td>
</tr>
<tr>
<td>Name[1].Language[1].Code</td>
<td>1</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td>1</td>
<td>2</td>
<td>en-gb</td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td>0</td>
<td>1</td>
<td>null</td>
</tr>
</tbody>
</table>
```json
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward;
  }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country;
    } optional string Url;
  }
}
```

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>R</th>
<th>D</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocId</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Links.Forward</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Links.Forward[1]</td>
<td>1</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Links.Forward[2]</td>
<td>1</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>Links.[Backward]</td>
<td>0</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>Name.Language.Code</td>
<td></td>
<td></td>
<td>en-us</td>
</tr>
<tr>
<td>Name.Language.Country</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>Name.Language[1].Code</td>
<td>2</td>
<td>2</td>
<td>en</td>
</tr>
<tr>
<td>Name.Language[1].Country</td>
<td>2</td>
<td>2</td>
<td>null</td>
</tr>
<tr>
<td>Name.Url</td>
<td></td>
<td></td>
<td><a href="http://a">http://a</a></td>
</tr>
<tr>
<td>Name[1].Language.[Code]</td>
<td>1</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>Name[1].Language.[Country]</td>
<td>1</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>Name[1].Url</td>
<td></td>
<td></td>
<td><a href="http://b">http://b</a></td>
</tr>
<tr>
<td>Name[2].Language.Code</td>
<td>1</td>
<td>2</td>
<td>en-gb</td>
</tr>
<tr>
<td>Name[2].Language.Country</td>
<td>1</td>
<td>3</td>
<td>gb</td>
</tr>
<tr>
<td>Name[2].[Url]</td>
<td></td>
<td>1</td>
<td>null</td>
</tr>
</tbody>
</table>

Apache: Big Data North America 2017
## Optimizations

### Projection Push Down

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>a4</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>a5</td>
<td>b5</td>
<td>c5</td>
</tr>
</tbody>
</table>

### Predicate Push Down

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>a4</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>a5</td>
<td>b5</td>
<td>c5</td>
</tr>
</tbody>
</table>

### Read Required Data

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>a4</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>a5</td>
<td>b5</td>
<td>c5</td>
</tr>
</tbody>
</table>

- Fetch required columns
- Filter records while reading
ENCODING

- Plain
- Dictionary Encoding
- Run Length Encoding/ Bit-Packing Hybrid
- Delta Encoding
- Delta-length byte array
- Delta Strings
COMPRESSION TRADE OFF

- CPU time
- Storage
- IO
- Bandwidth
Thank you

Ran.ga.na.than B
@ran_than

Apache: Big Data North America 2017