An Introduction to Parquet File Format for Data Analytics

Sawood Alam, Internet Archive
Mark Phillips, University of North Texas
April 24, 2024

What is Parquet?

Parquet is an open-source, **columnar** storage file format, designed for efficient and scalable data storage and retrieval, and was first released by Apache in 2013.

- Efficient data compression and encoding schemes
- Support for complex nested data structures
- Compatible with many data processing frameworks

Columnar Data Storage

	URL Key	DateTime	Status	Mime Type	Digest	Size
RG1	com,example)/	20020112142310	200	text/html	UY3I2DT2AMWAY6DECFCFYMT5ZOTFHUCH	458
	com,example)/	20020603042351	200	text/html	UY3I2DT2AMWAY6DECFCFYMT5ZOTFHUCH	462
	com,example)/	20110917022206	302	text/html	3I42H3S6NNFQ2MSVX7XZKYAYSCX5QBYJ	340
RG2	com,example)/	20230107230148	200	text/html	JI6OR3QR4CI526JD6TMMNZNV4QPMPQCH	1062
	com,example)/	20240226074219	-	warc/revisit	JI6OR3QR4CI526JD6TMMNZNV4QPMPQCH	541
	com,example)/robots.txt	20020719044147	200	text/plain	U6EHAWYE22F46AVZ7FUE7NPC7GP7GIXY	532
RG3	com,example)/robots.txt	20240219221238	200	text/plain	JI6OR3QR4CI526JD6TMMNZNV4QPMPQCH	528
	com,example)/xyz.html	20190523170632	404	text/html	B2LTWWPUOYAH7UIPQ7ZUPQ4VMBSVC36A	866

Parquet File Format

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
…
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
…
<Column N Chunk 2 + Column Metadata>
…
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
…
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata
4-byte magic number "PAR1"
```

Text vs. Binary Format

	Text	Binary
Examples	CDX[J], CSV, JSON	Parquet, SQLite
Storage/Memory/IO	Values are stored as strings of chars	Typed data may require fewer bytes
Access	Human-readable	Specialized tools needed
Errors	Usually tolerant to small errors	Small bitrots can corrupt the file
Longevity	Can be migrated to newer formats easily	Unpopular formats may lose tool support
Performance	Slower to process large data	Usually more efficient than text

Data Types

- **BOOLEAN**: 1 bit boolean
- **INT32**: 32 bit signed ints
- **INT64**: 64 bit signed ints
- **INT96**: 96 bit signed ints
- **FLOAT**: IEEE 32-bit floating point values
- **DOUBLE**: IEEE 64-bit floating point values
- **BYTE_ARRAY**: arbitrarily long byte arrays
- **FIXED_LEN_BYTE_ARRAY**: fixed length byte arrays

Encodings

- Plain
- Dictionary Encoding
- Run Length Encoding / Bit-Packing Hybrid
- Delta Encoding
- Delta-Length Byte Array
- Delta Strings
- Byte Stream Split

Compression

- **UNCOMPRESSED**: Data is left uncompressed
- **SNAPPY**: Snappy compression format
- **GZIP**: RFC 1952
- **LZO**: LZO compression library
- **BROTLI**: RFC 7932
- **ZSTD**: RFC 8478
- **LZ4_RAW**: LZ4 block format

Partitioning

- Arbitrary partition of data based on operator-chosen criteria
  - E.g., a fixed number of records per file or one file per day
  - Can be combined or repartitioned later
  - Allows distributed processing
  - Allows easier management of files
  - Scales well
- Multiple files can be queried using glob patterns
  - `SELECT COUNT(*) FROM '/data/{YEAR}/{MONTH}/*.parquet'`

Query Language Support

Data in Parquet files can be queried using multiple means:

- SQL
  - Similar to how SQL queries are performed on SQLite files
  - Some Parquet SQL query implementations allow glob patterns to query multiple files
- DataFrame
  - Popular DataFrame implementations like Pandas and Polars support reading/writing
  - Parallel processing of multiple files or streaming row-groups of large files is possible

Tools and Libraries Across Languages

- Parquet is a language-agnostic and self-descriptive format
- Many components of the Hadoop ecosystem support and embrace Parquet
- Many general-purpose languages like Python, Ruby, R, etc. have bindings with the Apache Arrow C++ implementation
- Many DataFrame libraries like Pandas and Polars have built-in functions to read and write Parquet

Import and Export from/to Other File Formats

- Usually a DataFrame implementation is used as an intermediary for conversion between Parquet and other structured data formats
  - CSV <> DataFrame <> Parquet
  - JSONL <> DataFrame <> Parquet
- To reduce memory requirements for large file conversion, streaming of data can be leveraged
  - When loading data into a DataFrame, max number of rows can be specified
  - Parquet files can be iterated one row group at a time
  - When writing large data to Parquet files, consider partitioning

Scripting Analytics Pipelines

- Ingest data and store them as Parquet files
  - The source of data can be stored files for batch processing or streams/logs
  - Partition data in temporally or any other condition as suitable
- Identify common useful questions and statistics about the data
- Write canned SQL queries or DataFrame functions to answer those questions
- Visualize statistics obtained from those queries
- Parametrize the script/dashboard to interact with the subset of data
- Consider means to drill down further with custom queries
- Incorporate anomaly detection and summarization options when suitable