Parquet File Format

parquet-java Classes Diagram

arnaud.nauwynck@gmail.com

ParquetFileReader public API [1/6] static methods

```
static ParquetMetadata readFooter(
        InputFile file, ParquetReadOptions options, SeekableInputStream f)


static ParquetFileReader open(
        InputFile file)

static ParquetFileReader open(
        InputFile file, ParquetReadOptions options)

static ParquetFileReader open(
        InputFile file, ParquetReadOptions options, SeekableInputStream f)
```

ParquetFileReader Public API [2/6] constructors

```
ParquetFileReader(
    Configuration conf, Path file, ParquetMetadata footer, ParquetReadOptions options)

ParquetFileReader(
    Configuration conf, Path file, ParquetMetadata footer, ParquetReadOptions options, SeekableInputStream f)

ParquetFileReader(
    InputFile file, ParquetReadOptions options)

ParquetFileReader(
    InputFile file, ParquetReadOptions options, SeekableInputStream f)
```

ParquetFileReader Public API [3/6] getter / misc..

```
String getFile()
ParquetMetadata getFooter()
FileMetaData getFileMetaData()
long getRecordCount()
long getFilteredRecordCount()
List<BlockMetaData> getRowGroups()

void setRequestedSchema(MessageType projection)

void appendTo(ParquetFileWriter writer)
```

ParquetFileReader Public API [4/6] read RowGroups

```
PageReadStore readNextRowGroup()
boolean skipNextRowGroup()
PageReadStore readRowGroup(int blockIndex)
PageReadStore readFilteredRowGroup(int blockIndex)
PageReadStore readNextFilteredRowGroup()
```

ParquetFileReader Public API [5/6] read RowGroup-Columns..

ColumnChunkPageReadStore readFilteredRowGroup(int blockIndex, RowRanges rowRanges)
ColumnIndexStore getColumnIndexStore(int blockIndex)
ColumnIndex readColumnIndex(ColumnChunkMetaData column)
OffsetIndex readOffsetIndex(ColumnChunkMetaData column)

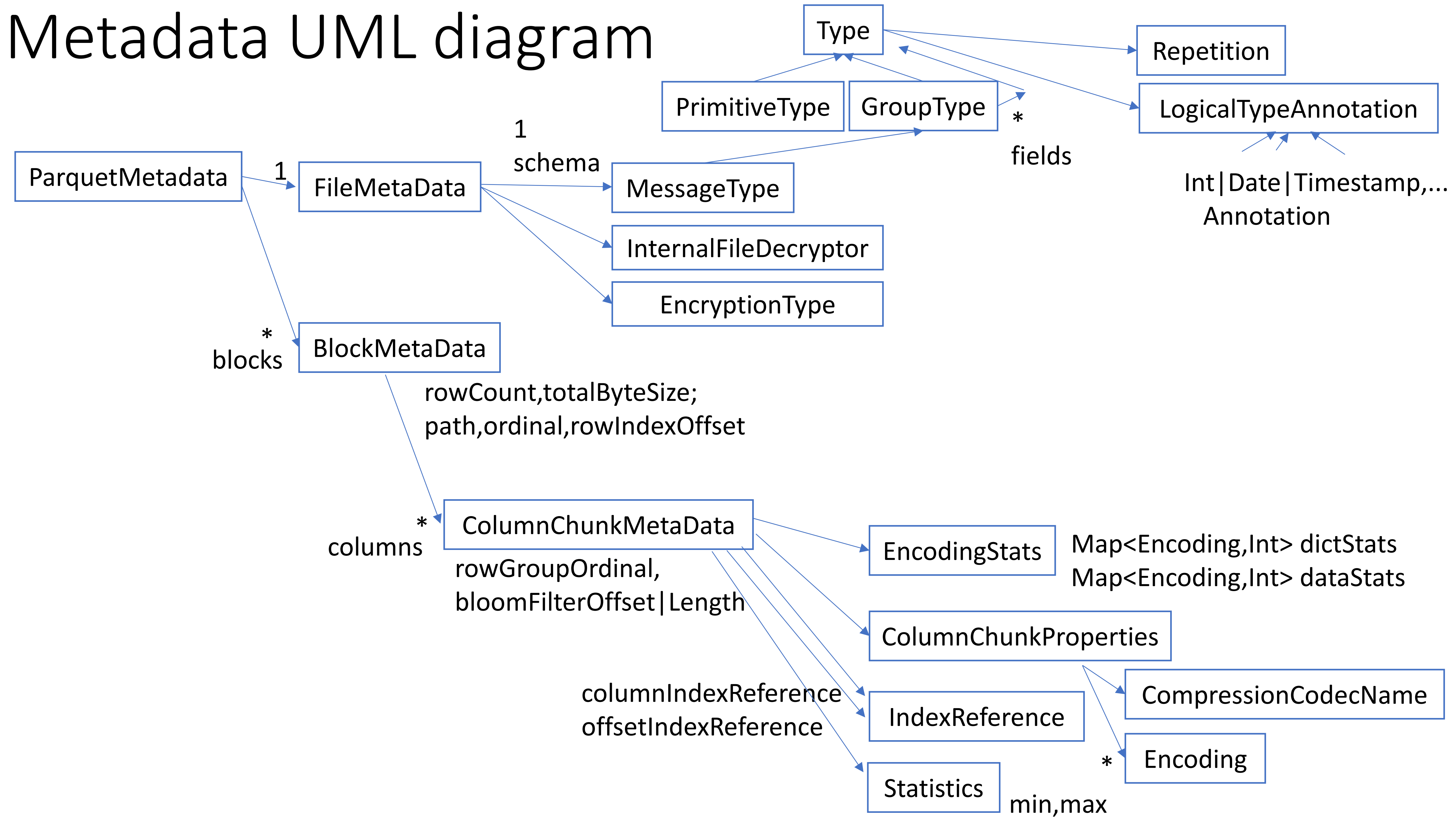
ParquetFileReader Public API [6/6] read DictionaryPage & BloomFilter

```
DictionaryPageReadStore getNextDictionaryReader()
DictionaryPageReader getDictionaryReader(int blockIndex)
DictionaryPageReader getDictionaryReader(BlockMetaData block)

BloomFilterReader getBloomFilterDataReader(int blockIndex)
BloomFilterReader getBloomFilterDataReader(BlockMetaData block)
BloomFilter readBloomFilter(ColumnChunkMetaData meta)
```

ParquetFileReader internal fields

```
public class ParquetFileReader implements Closeable {

  protected final SeekableInputStream f;
  private final InputFile file;
  private final ParquetReadOptions options;

  private final Map<ColumnPath, ColumnDescriptor> paths = new HashMap<>();
  private final FileMetaData fileMetaData;
  private final List<BlockMetaData> blocks;
  private final List<ColumnIndexStore> blockIndexStores;
  private final List<RowRanges> blockRowRanges;

  private ParquetMetadata footer;

  private int currentBlock;
  private ColumnChunkPageReadStore currentRowGroup;
  private DictionaryPageReader nextDictionaryReader;

  private InternalFileDecryptor fileDecryptor;
```

Metadata UML diagram



ParquetMetadata

```
public class ParquetMetadata {

  private final FileMetaData fileMetaData;
  private final List<BlockMetaData> blocks;

}
```

FileMetaData

```
public final class FileMetaData implements Serializable {

  private final MessageType schema;
  private final Map<String, String> keyValueMetaData;
  private final String createdBy;
  private final InternalFileDecryptor fileDecryptor;
  private final EncryptionType encryptionType;
```

MessageType (=schema)

```
public final class MessageType extends GroupType {
}
```

Type class & sub-classes

```
public abstract class Type {
  private final String name;
  private final Repetition repetition;
  private final LogicalTypeAnnotation logicalTypeAnnotation;
  private final ID id;
}

public class GroupType extends Type {
  private final List<Type> fields;
  private final Map<String, Integer> indexByName;
}

public final class PrimitiveType extends Type {
  private final PrimitiveTypeName primitive;
  private final int length;
  private final DecimalMetadata decimalMeta;
  private final ColumnOrder columnOrder;
}
```

ColumnDescriptor

```
public class ColumnDescriptor implements Comparable<ColumnDescriptor> {

  private final String[] path;
  private final PrimitiveType type;
  private final int maxRep;
  private final int maxDef;

}
```

Type internal

```
public static final class ID {
    private final int id;
}


public class ColumnOrder {
    private final ColumnOrderName columnOrderName;
}

public enum ColumnOrderName {
    UNDEFINED,
    TYPE_DEFINED_ORDER
  }
```

LogicalTypeAnnotation

```
public abstract class LogicalTypeAnnotation {

 enum LogicalTypeToken {
    MAP,
    LIST,
    STRING,
    MAP_KEY_VALUE,
    ENUM,
    DECIMAL,
    DATE,
    TIME,
    TIMESTAMP,
    INTEGER,
    JSON,
    BSON,
    UUID,
    INTERVAL,
    FLOAT16
}

}
```

classes extends LogicalTypeAnnotation

```
 public static class IntLogicalTypeAnnotation extends LogicalTypeAnnotation {
   private final int bitWidth;
   private final boolean isSigned;
}

class DecimalLogicalTypeAnnotation extends LogicalTypeAnnotation {
   private final PrimitiveStringifier stringifier;
   private final int scale;
   private final int precision;
}

 public static class TimestampLogicalTypeAnnotation extends LogicalTypeAnnotation {
   private final boolean isAdjustedToUTC;
   private final TimeUnit unit;
}

class TimeLogicalTypeAnnotation extends LogicalTypeAnnotation {
   private final boolean isAdjustedToUTC;
   private final TimeUnit unit;
}
```

PrimitiveTypeName

```
public static enum PrimitiveTypeName {

    INT64,     // long, signed/unsigned
    INT32,     // int, signed/unsigned
    BOOLEAN,
    BINARY,   // byte[]
    FLOAT,     //
    DOUBLE, //
    INT96,     //
    FIXED_LEN_BYTE_ARRAY  // byte[N]

}
```

Repetition

```
public enum Repetition {
   REQUIRED,   // exactly 1
   OPTIONAL,   // 0 or 1
   REPEATED    // 0 or more
};
```

InternalFileDecryptor

```
public class InternalFileDecryptor {

  private final FileDecryptionProperties fileDecryptionProperties;
  private final DecryptionKeyRetriever keyRetriever;
  private final boolean checkPlaintextFooterIntegrity;
  private final byte[] aadPrefixInProperties;
  private final AADPrefixVerifier aadPrefixVerifier;

  private byte[] footerKey;
  private HashMap<ColumnPath, InternalColumnDecryptionSetup> columnMap;
  private EncryptionAlgorithm algorithm;
  private byte[] fileAAD;
  private boolean encryptedFooter;
  private byte[] footerKeyMetaData;
  private boolean fileCryptoMetaDataProcessed = false;
  private BlockCipher.Decryptor aesGcmDecryptorWithFooterKey;
  private BlockCipher.Decryptor aesCtrDecryptorWithFooterKey;
  private boolean plaintextFile;
```

BlockMetaData

```
public class BlockMetaData {

  private List<ColumnChunkMetaData> columns;
  private long rowCount;
  private long totalByteSize;
  private String path;
  private int ordinal;
  private long rowIndexOffset;
}
```

ColumnChunkMetadata

```
public abstract class ColumnChunkMetaData {

  protected int rowGroupOrdinal = -1;

  EncodingStats encodingStats;
  ColumnChunkProperties properties;
  private IndexReference columnIndexReference;
  private IndexReference offsetIndexReference;
  private long bloomFilterOffset = -1;
  private int bloomFilterLength = -1;

  abstract Statistics getStatistics(); // cf sub-classes Statistics

}
```

IntColumnChunkMetaData

```
class IntColumnChunkMetaData extends ColumnChunkMetaData {

  private final int firstDataPage;
  private final int dictionaryPageOffset;
  private final int valueCount;
  private final int totalSize;
  private final int totalUncompressedSize;
  private final Statistics statistics;
  private final SizeStatistics sizeStatistics;
}
```

LongColumnChunkMetaData

```
class LongColumnChunkMetaData extends ColumnChunkMetaData {

  private final long firstDataPageOffset;
  private final long dictionaryPageOffset;
  private final long valueCount;
  private final long totalSize;
  private final long totalUncompressedSize;
  private final Statistics statistics;
  private final SizeStatistics sizeStatistics;
}
```

EncryptedColumnChunkMetaData

```
class EncryptedColumnChunkMetaData extends ColumnChunkMetaData {

  private final ParquetMetadataConverter parquetMetadataConverter;
  private final byte[] encryptedMetadata;
  private final byte[] columnKeyMetadata;
  private final InternalFileDecryptor fileDecryptor;

  private final int columnOrdinal;
  private final PrimitiveType primitiveType;
  private final String createdBy;
  private ColumnPath path;

  private boolean decrypted;
  private ColumnChunkMetaData shadowColumnChunkMetaData;
```

Statistics

```
public abstract class Statistics<T extends Comparable<T>> {

  private final PrimitiveType type;
  private final PrimitiveComparator<T> comparator;
  private boolean hasNonNullValue;
  private long num_nulls;
  final PrimitiveStringifier stringifier;

}
```

sub-classes:
{ NoOps | Binary | Boolean | Int | Long [Float | Double } Statistics

{Binary|Float| ... }Statistics

```
class BinaryStatistics extends Statistics<Binary> {   private Binary min, max;}

abstract class Binary {
  protected boolean isBackingBytesReused;
  public abstract byte][ getBytes();
}

class BooleanStatistics extends Statistics<Float> {   private boolean min, max;  }
class IntStatistics extends Statistics<Float> {   private int min, max;  }
class LongStatistics extends Statistics<Float> {   private long min, max;  }
class FloatStatistics extends Statistics<Float> {   private float min, max;  }
class DoubleStatistics extends Statistics<Float> {   private double min, max;  }

class NoopsStatistics extends Statistics<Float> {  }
```

SizeStatistics

```
public class SizeStatistics {

  private final PrimitiveType type;
  private long unencodedByteArrayDataBytes;
  private final List<Long> repetitionLevelHistogram;
  private final List<Long> definitionLevelHistogram;

  private boolean valid = true;
}
```

EncodingStats

```
public class EncodingStats {

  final Map<Encoding, Number> dictStats;
  final Map<Encoding, Number> dataStats;
  private final boolean usesV2Pages;
}
```

Encoding

```
public enum Encoding {

  PLAIN,
  RLE,
  BYTE_STREAM_SPLIT,
  @Deprecated BIT_PACKED,
  @Deprecated PLAIN_DICTIONARY,
  DELTA_BINARY_PACKED,
  DELTA_LENGTH_BYTE_ARRAY,
  DELTA_BYTE_ARRAY,
  RLE_DICTIONARY
  ;
}
```

ColumnChunkProperties

```
public class ColumnChunkProperties {

  private final CompressionCodecName codec;
  private final ColumnPath path;
  private final PrimitiveType type;
  private final Set<Encoding> encodings;

}
```

CompressionCodecName

```
public enum CompressionCodecName {

  UNCOMPRESSED,
  SNAPPY,
  GZIP,
  LZO,
  BROTLI,
  LZ4,
  ZSTD,
  LZ4_RAW;
}
```

ColumnPath

```
public final class ColumnPath implements Iterable<String>, Serializable {

  private final String[] p;

}
```

RowRanges - Range

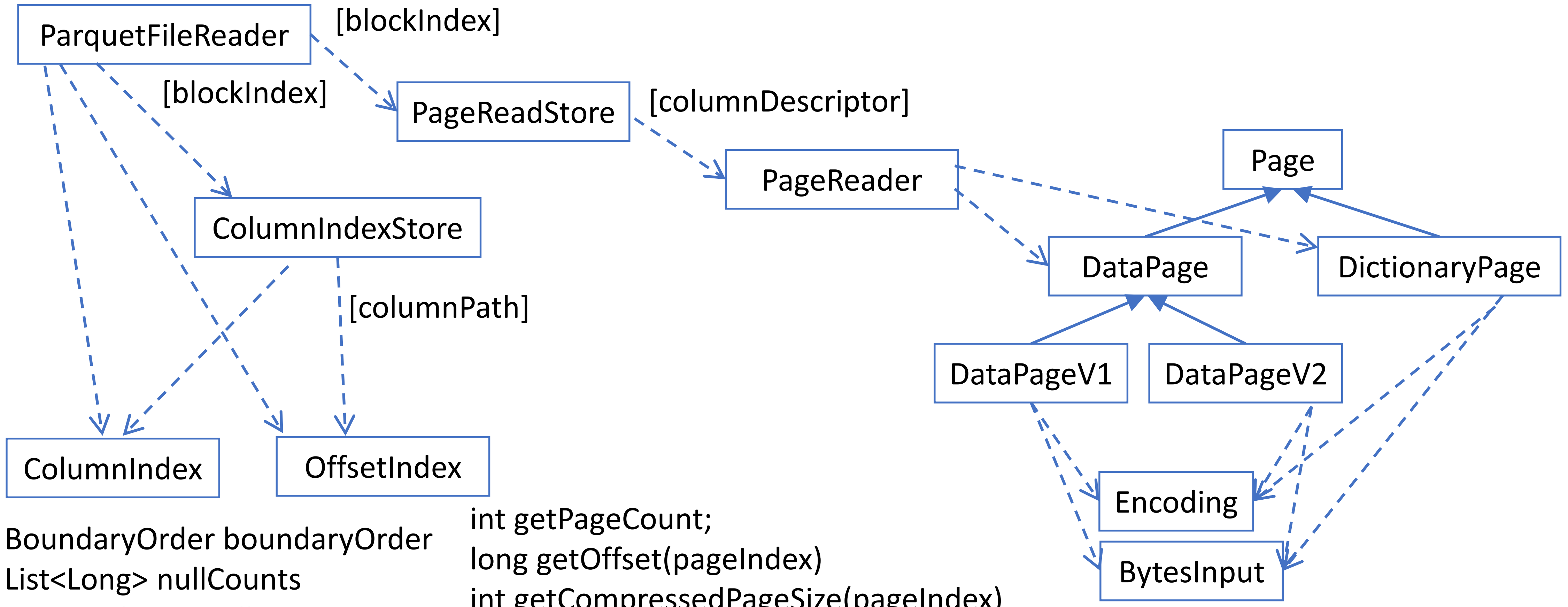
```
public class ParquetFileReader implements Closeable {
  ...
  private final List<RowRanges> blockRowRanges;
}


public class RowRanges {
   private final List<Range> ranges;
}

public class Range {
   public final long from;
   public final long to;
}
```

ColumnIndexStore

```
public class ParquetFileReader implements Closeable {
   ...
   private final List<ColumnIndexStore> blockIndexStores;
}


public interface ColumnIndexStore {
     ColumnIndex getColumnIndex(ColumnPath column);
     OffsetIndex getOffsetIndex(ColumnPath column);
}
```

RowGroups - Chunk - Pages

ParquetFile Read API - UML Classes



int getPageCount;
long getOffset(pageIndex)
int getCompressedPageSize(pageIndex)
long getFirstRowIndex(pageIndex)

BoundaryOrder boundaryOrder
List<Long> nullCounts
List<Boolean> nullPages
List<ByteBuffer> minValues
List<ByteBuffer> maxValues
List<Long> repetitionLevelHistogram
List<Long> definitionLevelHistogram

PageReadStore
(.. from ParquetFileReader read API)

```
class ParquetFileReader {
..

PageReadStore readNextRowGroup()
boolean skipNextRowGroup()
PageReadStore readRowGroup(int blockIndex)
PageReadStore readFilteredRowGroup(int blockIndex)
PageReadStore readNextFilteredRowGroup()

}
```

PageReadStore

```
/**
 * contains all the readers for all the columns of the corresponding row group
 * <p>
 * TODO: rename to RowGroup?
 */
public interface PageReadStore extends AutoCloseable {

  PageReader getPageReader(ColumnDescriptor descriptor);

  long getRowCount();
  default Optional<Long> getRowIndexOffset()
  default Optional<PrimitiveIterator.OfLong> getRowIndexes()
}
```

PageReader

```
/**
 * Reader for a sequence a page from a given column chunk
 */
public interface PageReader {

  DictionaryPage readDictionaryPage();

  long getTotalValueCount();

 /**
   * @return the next page in that chunk or null if after the last page
   */
  DataPage readPage();
}
```

Page

```
/**
 * one page in a chunk
 */
public abstract class Page {

  private final int compressedSize;
  private final int uncompressedSize;
}
```

DataPage

```
/**
 * one data page in a chunk
 */
public abstract class DataPage extends Page {

  private final int valueCount;
  private final long firstRowIndex;

}
```

```
public class DataPageV1 extends DataPage {

  private final BytesInput bytes;
  private final Statistics<?> statistics;
  private final Encoding rlEncoding;
  private final Encoding dlEncoding;
  private final Encoding valuesEncoding;
  private final int indexRowCount;
}
```

```
public class DataPageV2 extends DataPage {

  private final int rowCount;
  private final int nullCount;
  private final BytesInput repetitionLevels;
  private final BytesInput definitionLevels;
  private final Encoding dataEncoding;
  private final BytesInput data;
  private final Statistics<?> statistics;
  private final boolean isCompressed;
```

DictionaryPage

```
public class DictionaryPage extends Page {

  private final BytesInput bytes;
  private final int dictionarySize;
  private final Encoding encoding;

}
```

ColumnChunkPageReadStore (... from ParquetFileReader read API)

```
class ParquetFileReader {
...
ColumnChunkPageReadStore readFilteredRowGroup(int blockIndex, RowRanges rowRanges)
}
```

```
/** package protected ?????!!*/
class ColumnChunkPageReadStore implements PageReadStore, DictionaryPageReadStore {
..
@Override ...

}
```

ColumnIndexStore
(.. from ParquetFileReader read API)

```
class ParquetFileReader {   ...
  ColumnIndexStore getColumnIndexStore(int blockIndex)
  ColumnIndex readColumnIndex(ColumnChunkMetaData column)
  OffsetIndex readOffsetIndex(ColumnChunkMetaData column)
}

public interface ColumnIndexStore {
    ColumnIndex getColumnIndex(ColumnPath column);
    OffsetIndex getOffsetIndex(ColumnPath column);
}
```

ColumnIndex

```
/**
 * Column index containing min/max and null count values for the pages in a column chunk.
 */
public interface ColumnIndex extends Visitor<PrimitiveIterator.OfInt> {

  public BoundaryOrder getBoundaryOrder();
  public List<Long> getNullCounts();
  public List<Boolean> getNullPages();
  public List<ByteBuffer> getMinValues();
  public List<ByteBuffer> getMaxValues();

  default List<Long> getRepetitionLevelHistogram();
  default List<Long> getDefinitionLevelHistogram();
}
```

OffsetIndex

```
/**
 * Offset index containing the offset and size of the page and the index of the first row in the page.
*/
public interface OffsetIndex {

  public int getPageCount();
  public long getOffset(int pageIndex);
  public int getCompressedPageSize(int pageIndex);
  public long getFirstRowIndex(int pageIndex);

  public default int getPageOrdinal(int pageIndex);
  public default long getLastRowIndex(int pageIndex, long rowGroupRowCount);
  public default Optional<Long> getUnencodedByteArrayDataBytes(int pageIndex);
}
```

DictionaryPageReadStore (.. from ParquetFileReader read API)

```
class ParquetFileReader { ....
 DictionaryPageReadStore getNextDictionaryReader()
 DictionaryPageReader getDictionaryReader(int blockIndex)
 DictionaryPageReader getDictionaryReader(BlockMetaData block)
```

```
/**
 * Interface to read dictionary pages for all the columns of a row group
 */
public interface DictionaryPageReadStore extends AutoCloseable {
   DictionaryPage readDictionaryPage(ColumnDescriptor descriptor);
}
```

```
/* package protected ???!!!! */
class DictionaryPageReader implements DictionaryPageReadStore {
   @Override ..
}
```

BloomFilterReader
(... From ParquetFileReader)

```
class ParquetFileReader {...
  BloomFilterReader getBloomFilterDataReader(int blockIndex)
  BloomFilterReader getBloomFilterDataReader(BlockMetaData block)
  BloomFilter readBloomFilter(ColumnChunkMetaData meta)
```

```
public class BloomFilterReader {

  public BloomFilter readBloomFilter(ColumnChunkMetaData meta)

}
```

BloomFilter

```
public interface BloomFilter {

void insertHash(long hash);
boolean findHash(long hash);
int getBitsetSize();

long hash(int value);    long hash(long value);  long hash(double value);
long hash(float value); long hash(Binary value); long hash(Object value);

HashStrategy getHashStrategy();
Algorithm getAlgorithm();
Compression getCompression();
default boolean canMergeFrom(BloomFilter otherBloomFilter)
default void merge(BloomFilter otherBloomFilter)

}
```