LanguageManual ORC

ORC Files

- ORC Files
  - ORC File Format
    - File Structure
    - Stripe Structure
  - HiveQL Syntax
  - Serialization and Compression
    - Integer Column Serialization
    - String Column Serialization
    - Compression
  - ORC File Dump Utility
  - ORC Configuration Parameters
- ORC Format Specification

ORC File Format

> **Version**
>
> Introduced in Hive version 0.11.0.

The *Optimized Row Columnar* (ORC) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using ORC files improves performance when Hive is reading, writing, and processing data.

Compared with RCFile format, for example, ORC file format has many advantages such as:

- a single file as the output of each task, which reduces the NameNode's load
- Hive type support including datetime, decimal, and the complex types (struct, list, map, and union)
- light-weight indexes stored within the file
  - skip row groups that don't pass predicate filtering
  - seek to a given row
- block-mode compression based on data type
  - run-length encoding for integer columns
  - dictionary encoding for string columns
- concurrent reads of the same file using separate RecordReaders
- ability to split files without scanning for markers
- bound the amount of memory needed for reading or writing
- metadata stored using Protocol Buffers, which allows addition and removal of fields

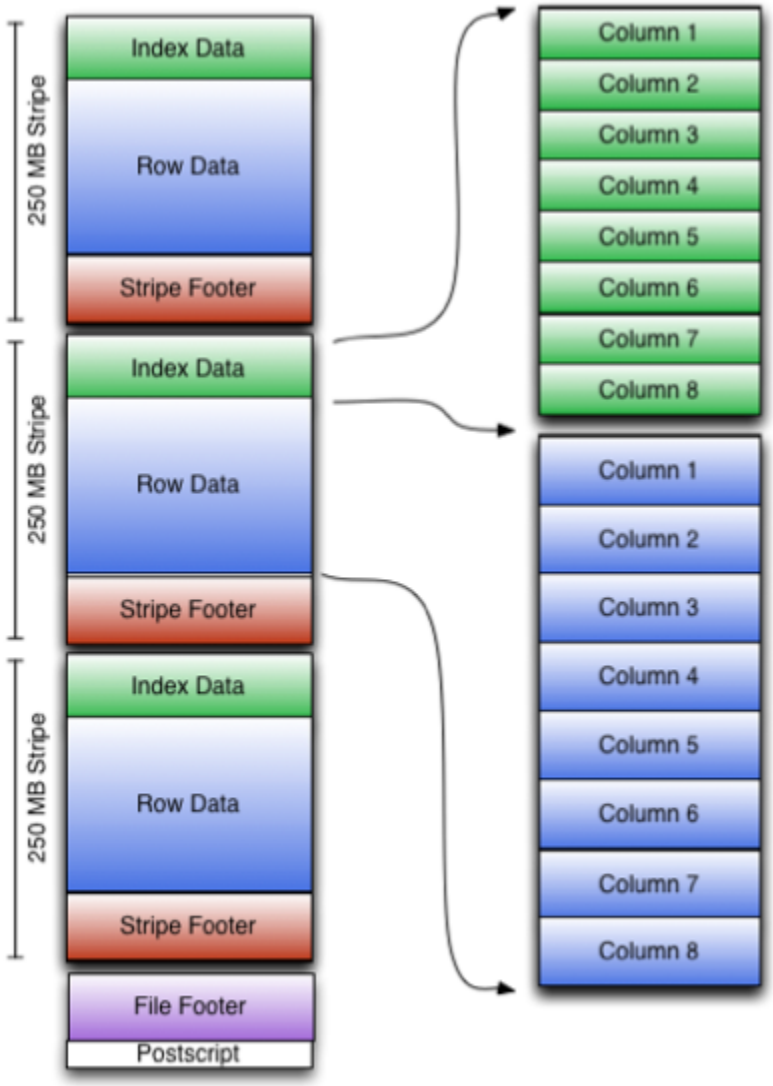
File Structure

An ORC file contains groups of row data called **stripes**, along with auxiliary information in a **file footer**. At the end of the file a **postscript** holds compression parameters and the size of the compressed footer.

The default stripe size is 250 MB. Large stripe sizes enable large, efficient reads from HDFS.

The file footer contains a list of stripes in the file, the number of rows per stripe, and each column's data type. It also contains column-level aggregates count, min, max, and sum.

This diagram illustrates the ORC file structure:



Stripe Structure

As shown in the diagram, each stripe in an ORC file holds index data, row data, and a stripe footer.

The **stripe footer** contains a directory of stream locations. **Row data** is used in table scans.

Index data includes min and max values for each column and the row positions within each column. (A bit field or bloom filter could also be included.) Row index entries provide offsets that enable seeking to the right compression block and byte within a decompressed block. Note that ORC indexes are used only for the selection of stripes and row groups and not for answering queries.

Having relatively frequent row index entries enables row-skipping within a stripe for rapid reads, despite large stripe sizes. By default every 10,000 rows can be skipped.

With the ability to skip large sets of rows based on filter predicates, you can sort a table on its secondary keys to achieve a big reduction in execution time. For example, if the primary partition is transaction date, the table can be sorted on state, zip code, and last name. Then looking for records in one state will skip the records of all other states.

A complete specification of the format is given in the ORC specification.

HiveQL Syntax

File formats are specified at the table (or partition) level. You can specify the ORC file format with HiveQL statements such as these:

- `CREATE TABLE ... STORED AS ORC`
- `ALTER TABLE ... [PARTITION partition_spec] SET FILEFORMAT ORC`
- `SET hive.default.fileformat=Orc`

The parameters are all placed in the TBLPROPERTIES (see Create Table). They are:

Key	Default	Notes
orc.compress	ZLIB	high level compression (one of NONE, ZLIB, SNAPPY)
orc.compress.size	262,144	number of bytes in each compression chunk
orc.stripe.size	67,108,864	number of bytes in each stripe
orc.row.index.stride	10,000	number of rows between index entries (must be >= 1000)
orc.create.index	true	whether to create row indexes
orc.bloom.filter.columns	""	comma separated list of column names for which bloom filter should be created
orc.bloom.filter.fpp	0.05	false positive probability for bloom filter (must >0.0 and <1.0)

For example, creating an ORC stored table without compression:

```
create table Addresses (
  name string,
  street string,
  city string,
  state string,
  zip int
) stored as orc tblproperties ("orc.compress"="NONE");
```

Version 0.14.0+: CONCATENATE

`ALTER TABLE table_name [PARTITION partition_spec] CONCATENATE` can be used to merge small ORC files into a larger file, starting in Hive 0.14.0. The merge happens at the stripe level, which avoids decompressing and decoding the data.

Serialization and Compression

The serialization of column data in an ORC file depends on whether the data type is integer or string.

Integer Column Serialization

Integer columns are serialized in two streams.

1. *present* bit stream: is the value non-null?
2. data stream: a stream of integers

Integer data is serialized in a way that takes advantage of the common distribution of numbers:

- Integers are encoded using a variable-width encoding that has fewer bytes for small integers.
- Repeated values are run-length encoded.
- Values that differ by a constant in the range (-128 to 127) are run-length encoded.

The *variable-width encoding* is based on Google's protocol buffers and uses the high bit to represent whether this byte is not the last and the lower 7 bits to encode data. To encode negative numbers, a zigzag encoding is used where 0, -1, 1, -2, and 2 map into 0, 1, 2, 3, 4, and 5 respectively.

Each set of numbers is encoded this way:

- If the first byte (b0) is negative:
  - -b0 variable-length integers follow.
- If the first byte (b0) is positive:
  - it represents b0 + 3 repeated integers
  - the second byte (-128 to +127) is added between each repetition
  - 1 variable-length integer.

In *run-length encoding,* the first byte specifies run length and whether the values are literals or duplicates. Duplicates can step by -128 to +128. Run-length encoding uses protobuf style variable-length integers.

String Column Serialization

Serialization of string columns uses a dictionary to form unique column values. The dictionary is sorted to speed up predicate filtering and improve compression ratios.

String columns are serialized in four streams.

1. *present* bit stream: is the value non-null?
2. dictionary data: the bytes for the strings
3. dictionary length: the length of each entry
4. row data: the row values

Both the dictionary length and the row values are run-length encoded streams of integers.

Compression

Streams are compressed using a codec, which is specified as a table property for all streams in that table. To optimize memory use, compression is done incrementally as each block is produced. Compressed blocks can be jumped over without first having to be decompressed for scanning. Positions in the stream are represented by a block start location and an offset into the block.

The codec can be Snappy, Zlib, or *none*.

ORC File Dump Utility

The ORC file dump utility analyzes ORC files. To invoke it, use this command:

```
// Hive version 0.11 through 0.14:
hive --orcfiledump <location-of-orc-file>

// Hive version 1.1.0 and later:
hive --orcfiledump [-d] [--rowindex <col_ids>] <location-of-orc-file>

// Hive version 1.2.0 and later:
hive --orcfiledump [-d] [-t] [--rowindex <col_ids>] <location-of-orc-file>

// Hive version 1.3.0 and later:
hive --orcfiledump [-j] [-p] [-d] [-t] [--rowindex <col_ids>] [--recover] [--skip-dump]
    [--backup-path <new-path>] <location-of-orc-file-or-directory>
```

Specifying `-d` in the command will cause it to dump the ORC file data rather than the metadata (Hive 1.1.0 and later).

Specifying `--rowindex` with a comma separated list of column ids will cause it to print row indexes for the specified columns, where 0 is the top level struct containing all of the columns and 1 is the first column id (Hive 1.1.0 and later).

Specifying `-t` in the command will print the timezone id of the writer.

Specifying `-j` in the command will print the ORC file metadata in JSON format. To pretty print the JSON metadata, add `-p` to the command.

Specifying `--recover` in the command will recover a corrupted ORC file generated by Hive streaming.

Specifying `--skip-dump` along with `--recover` will perform recovery without dumping metadata.

Specifying `--backup-path` with a *new-path* will let the recovery tool move corrupted files to the specified backup path (default: /tmp).

<*location-of-orc-file*> is the URI of the ORC file.

<*location-of-orc-file-or-directory*> is the URI of the ORC file or directory. From Hive 1.3.0 onward, this URI can be a directory containing ORC files.

ORC Configuration Parameters

The ORC configuration parameters are described in Hive Configuration Properties – ORC File Format.

ORC Format Specification

The ORC specification has moved to ORC project.