*ORC Schema Evolution Issues (Vectorized, ACID, and Non-Vectorized)*

*HIVE-11981*

v 0.1 September 28th, 2015

# Introduction

This change fixes high priority issues with schema evolution for the ORC file format.

Schema evolution here is limited to adding new columns and a few cases of column type-widening (e.g. int to bigint).

Renaming columns, deleting column, moving columns and other schema evolution were not pursued due to lack of importance and lack of time. Also, it appears a much more sophisticated metadata would be needed to support them.

The biggest issues for users have been adding new columns for ACID table (HIVE-11421 Support Schema evolution for ACID tables) and vectorization (HIVE-10598 Vectorization borks when column is added to table).

Major issues shows up when writing new data based on the (global) table schema into partitions with older table schemas. The bottom line is if users need to do this they need to keep all partition table metadata the same. That is, when they do schema evolution with ALTER TABLE they must include the CASCADE clause so table schema changes are propagated to all partitions.

# ACID (Execution Errors)

ACID tables suffer execution error (index out of bounds) when merging base file and deltas with different schemas.

ACID merging is in the ORC layer and needs to remain there. So we realized that converting data with old schemas must be done a low level in ORC.

The change adds the following to the ReaderImpl:

1. If ORC is being called from a MapWork vertex, find the PartitionDesc for the ORC file path and extract the partition’s table column names and types from the metadata properties.
2. If not, then, see if the configuration contains table column names and types.
3. Fall back to using the file’s schema.

ACID will be case #1.

Any ACID base files and delta data will be converted to the partition’s table schema and will merge correctly.

WITH ONE BIG EXCEPTION.

When new data is written into Hive tables, it uses the (global) table’s schema not the partition’s table schema.

So, if the partition’s table schema is old then merging will either fail or clip off data in new columns.

Merging will fail if it encounters a new ORC file in the partition with types wider than the partition’s table schema. Why? The new code looks at the new file’s table schema and sees that is cannot convert an Int to a SmallInt, for example.

Or, if there are new columns in the new data that data will not be read during regular queries, by update statements, or during merging and will not be seen. This is probably not what the user really wants.

IMPLICATION: We should always require that ACID Schema Evolution update the table schema in all partitions. That is, require the use the CASCADE clause when issuing ALTER TABLE.

What is unclear is what to do about concurrent ALTER TABLE and new writes.

# ORC Partition Table Mixing (Execution Errors)

The same restriction we propose for ACID Tables – requiring that Schema Evolution update the table schema in all partitions – is needed if new data is written to old partitions.

# ORC Non-Partition Table Mixing (Execution Errors)

The changes above for ACID Tables that use the PartitionDesc or table column name and types with data conversion will allow an ORC Non-Partitioned Table to read without execution errors.

# Vectorization (Metadata Mismatch Causing Execution Errors)

The high performance of Vectorization is based a batch of 1024 rows (VectorizedRowBatch).

The batch is allocated once and its storage is reused. This avoids the high Java object allocation and garbage collection overhead of per field objects in the regular execution engine.

Also, when an operator produces new column values they are generated into extra columns of the batch (called scratch columns) instead of producing a new batch.

Here is an example VectorizedRowBatch with 9 columns. Operators avoid copying by projecting the columns they want out of the batch.

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | \_col0 | \_col1 | \_col2 |

4 scratch columns

5 table columns

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TYPE | tinyint | smallint | int | double | string | int | double | tinyint | double |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NAME | ctinyint | csmallint | cint | cdouble | cstring1 |

Initial (table) projection.

e.g. Intermediate operator projection.

Anyway, the issue here is that the vectorized operators expects the VectorizedRowBatch returned by input formats such as ORC to be in the **one** Vectorization column schema based on the (global) table schema and the scratch columns needed by the vectorized operator tree.

Today, the ORC format creates a VectorizedRowBatch based on the *partition’s* table metadata which can be different due to schema evolution than the overall table.

Execution errors like cast mismatch or null pointer exception result.

Changes include:

1. Have the Vectorizer class create one global VectorizedRowBatchCtx (context) for each MapWork or ReduceWork vertex.
2. Make the VectorizedRowBatchCtx object serializable and usable for both creating VectorizedRowBatch and dealing with partitioning.
3. Make the ORC input format use the one global MapWork VectorizedRowBatchCtx object to create the VectorizedRowBatch that it will fill in.

A major assumption in Vectorization is that table columns are always in the same order (i.e no column deletions, renames, moves, etc). Also, that only the number of columns in a partition’s metadata can be fewer and/or in limited cases a smaller type.

Continuing the earlier example if a partition’s table metadata only had 3 table columns, the last 2 table columns need to be defaulted to NULL.

Filled from ORC columns Not Touched by ORC.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TYPE | tinyint | smallint | int | double | string | int | double | tinyint | double |

Defaulted to NULL by ORC.

ORC gets the global table (non-partitioned) column count from MapWork’s VectorizedRowBatchCtx so it can figure out how many columns to default to NULL.

# Vectorization (Allow Type Widening)

A temporary fix was made to Vectorization to not vectorize queries that accessed any partition with table metadata different than the table description needed the MapWork TableScanOperator. This prevented execution errors but prevented queries from running in high performance vectorization mode.

Changes include:

1. Enhance the Vectorizer to vectorize when the number of partition table schema columns varies and when there ORC’s limited type-widening changes.