

# StatsDev

## Statistics in Hive

- Statistics in Hive
  - Motivation
  - Scope
    - Table and Partition Statistics
    - Column Statistics
    - Top K Statistics
  - Quick overview
  - Implementation
  - Usage
    - Configuration Variables
    - Newly Created Tables
    - Existing Tables – ANALYZE
  - Examples
    - ANALYZE TABLE <table1> CACHE METADATA
  - Current Status (JIRA)

This document describes the support of statistics for Hive tables (see [HIVE-33](#)).

### Motivation

Statistics such as the number of rows of a table or partition and the histograms of a particular interesting column are important in many ways. One of the key use cases of statistics is query optimization. Statistics serve as the input to the cost functions of the optimizer so that it can compare different plans and choose among them. Statistics may sometimes meet the purpose of the users' queries. Users can quickly get the answers for some of their queries by only querying stored statistics rather than firing long-running execution plans. Some examples are getting the quantile of the users' age distribution, the top 10 apps that are used by people, and the number of distinct sessions.

### Scope

#### Table and Partition Statistics

The first milestone in supporting statistics was to support table and partition level statistics. Table and partition statistics are now stored in the Hive Metastore for either newly created or existing tables. The following statistics are currently supported for partitions:

- Number of rows
- Number of files
- Size in Bytes

For tables, the same statistics are supported with the addition of the number of partitions of the table.



#### Version: Table and partition statistics

Table and partition level statistics were added in Hive 0.7.0 by [HIVE-1361](#).

#### Column Statistics

The second milestone was to support column level statistics. See [Column Statistics in Hive](#) in the Design Documents.

Supported column stats are:

BooleanColumnStatsData	DoubleColumnStatsData	LongColumnStatsData	StringColumnStatsData	BinaryColumnStatsData	DecimalColumnStatsData	Date	DateColumnStatsData	Timestamp	TimestampColumnStatsData	union ColumnStatisticsData
1: required i64 numTrues,	1: optional double lowValue,	1: optional i64 lowValue,	1: required i64 maxColLen,	1: required i64 maxColLen,	1: optional Decimal lowValue,	1: required i64 daysSinceEpoch	1: optional Date lowValue,	1: required i64 secondsSinceEpoch	1: optional Timestamp lowValue,	1: BooleanColumnStatsData booleanStats,
2: required i64 numFalses,	2: optional double highValue,	2: optional i64 highValue,	2: required double avgColLen,	2: required double avgColLen,	2: optional Decimal highValue,		2: optional Date highValue,		2: optional Timestamp highValue,	2: LongColumnStatsData longStats,
3: required i64 numNulls,	3: required i64 numNulls,	3: required i64 numNulls,	3: required i64 numNulls,	3: required i64 numNulls,	3: required i64 numNulls,		3: required i64 numNulls,		3: required i64 numNulls,	3: DoubleColumnStatsData doubleStats,
4: optional binary bitVectors	4: required i64 numDVs,	4: required i64 numDVs,	4: required i64 numDVs,	4: optional binary bitVectors	4: required i64 numDVs,		4: required i64 numDVs,		4: required i64 numDVs,	4: StringColumnStatsData stringStats,

	5: optional binary bitVectors,	5: optional binary bitVectors,	5: optional binary bitVectors		5: optional binary bitVectors,		5: optional binary bitVectors,		5: optional binary bitVectors,	5: BinaryColumnStatsData binaryStats,
	6: optional binary histogram	6: optional binary histogram			6: optional binary histogram		6: optional binary histogram		6: optional binary histogram	6: DecimalColumnStatsData decimalStats,
										7: DateColumnStatsData dateStats,
										8: TimestampColumnStatsData timestampStats



#### Version: Column statistics

Column level statistics were added in Hive 0.10.0 by [HIVE-1362](#).

## Top K Statistics

Column level top K statistics are still pending; see [HIVE-3421](#).

## Quick overview

Description	Stored in	Collected by	Since
Number of partition the dataset consists of	Fictional metastore property: <b>numPartitions</b>	computed during displaying the properties of a partitioned table	<a href="#">Hive 2.3</a>
Number of files the dataset consists of	Metastore table property: <b>numFiles</b>	Automatically during Metastore operations	
Total size of the dataset as its seen at the filesystem level	Metastore table property: <b>totalSize</b>		
Uncompressed size of the dataset	Metastore table property: <b>rawDataSize</b>	Computed, these are the basic statistics. Calculated automatically when <a href="#">hive.stats.autogather</a> is enabled. Can be collected manually by: ANALYZE TABLE ... COMPUTE STATISTICS	<a href="#">Hive 0.8</a>
Number of rows the dataset consist of	Metastore table property: <b>numRows</b>		
Column level statistics	Metastore; TAB_COL_STATS table	Computed, Calculated automatically when <a href="#">hive.stats.column.autogather</a> is enabled. Can be collected manually by: ANALYZE TABLE ... COMPUTE STATISTICS FOR COLUMNS	

## Implementation

The way the statistics are calculated is similar for both newly created and existing tables.

For newly created tables, the job that creates a new table is a MapReduce job. During the creation, every mapper while copying the rows from the source table in the FileSink operator, gathers statistics for the rows it encounters and publishes them into a Database (possibly [MySQL](#)). At the end of the MapReduce job, published statistics are aggregated and stored in the MetaStore.

A similar process happens in the case of already existing tables, where a Map-only job is created and every mapper while processing the table in the TableScan operator, gathers statistics for the rows it encounters and the same process continues.

It is clear that there is a need for a database that stores temporary gathered statistics. Currently there are two implementations, one is using [MySQL](#) and the other is using [HBase](#). There are two pluggable interfaces [IStatsPublisher](#) and [IStatsAggregator](#) that the developer can implement to support any other storage. The interfaces are listed below:

```

package org.apache.hadoop.hive ql.stats;

import org.apache.hadoop.conf.Configuration;

/**
 * An interface for any possible implementation for publishing statics.
 */

public interface IStatsPublisher {

    /**
     * This method does the necessary initializations according to the implementation requirements.
     */
    public boolean init(Configuration hconf);

    /**
     * This method publishes a given statistic into a disk storage, possibly HBase or MySQL.
     *
     * rowID : a string identification the statistics to be published then gathered, possibly the table name + the
     partition specs.
     *
     * key : a string noting the key to be published. Ex: "numRows".
     *
     * value : an integer noting the value of the published key.
     */
    public boolean publishStat(String rowID, String key, String value);

    /**
     * This method executes the necessary termination procedures, possibly closing all database connections.
     */
    public boolean terminate();
}

```

```

package org.apache.hadoop.hive ql.stats;

import org.apache.hadoop.conf.Configuration;

/**
 * An interface for any possible implementation for gathering statistics.
 */

public interface IStatsAggregator {

    /**
     * This method does the necessary initializations according to the implementation requirements.
     */
    public boolean init(Configuration hconf);

    /**
     * This method aggregates a given statistic from a disk storage.
     * After aggregation, this method does cleaning by removing all records from the disk storage that have the
     same given rowID.
     *
     * rowID : a string identification the statistic to be gathered, possibly the table name + the partition specs.
     *
     * key : a string noting the key to be gathered. Ex: "numRows".
     *
     */
    public String aggregateStats(String rowID, String key);

    /**
     * This method executes the necessary termination procedures, possibly closing all database connections.
     */
    public boolean terminate();
}

```

## Usage

### Configuration Variables

See [Statistics](#) in [Configuration Properties](#) for a list of the variables that configure Hive table statistics. [Configuring Hive](#) describes how to use the variables.

### Newly Created Tables

For newly created tables and/or partitions (that are populated through the [INSERT OVERWRITE](#) command), statistics are automatically computed by default. The user has to explicitly set the boolean variable [hive.stats.autogather](#) to **false** so that statistics are not automatically computed and stored into Hive MetaStore.

```
set hive.stats.autogather=false;
```

The user can also specify the implementation to be used for the storage of temporary statistics setting the variable [hive.stats.dbclass](#). For example, to set HBase as the implementation of temporary statistics storage (the default is `jdbc:derby` or `fs`, depending on the Hive version) the user should issue the following command:

```
set hive.stats.dbclass=hbase;
```

In case of JDBC implementations of temporary stored statistics (ex. Derby or MySQL), the user should specify the appropriate connection string to the database by setting the variable [hive.stats.dbconnectionstring](#). Also the user should specify the appropriate JDBC driver by setting the variable [hive.stats.jdbcdriver](#).

```

set hive.stats.dbclass=jdbc:derby;
set hive.stats.dbconnectionstring="jdbc:derby:;databaseName=TempStatsStore;create=true";
set hive.stats.jdbcdriver="org.apache.derby.jdbc.EmbeddedDriver";

```

Queries can fail to collect stats completely accurately. There is a setting [hive.stats.reliable](#) that fails queries if the stats can't be reliably collected. This is false by default.

## Existing Tables – ANALYZE

For existing tables and/or partitions, the user can issue the ANALYZE command to gather statistics and write them into Hive MetaStore. The syntax for that command is described below:

```
ANALYZE TABLE [db_name.]tablename [PARTITION(partcol1[=val1], partcol2[=val2], ...)] -- (Note: Fully support
qualified table name since Hive 1.2.0, see HIVE-10007.)
  COMPUTE STATISTICS
  [FOR COLUMNS]           -- (Note: Hive 0.10.0 and later.)
  [CACHE METADATA]        -- (Note: Hive 2.1.0 and later.)
  [NOSCAN];
```

When the user issues that command, he may or may not specify the partition specs. If the user doesn't specify any partition specs, statistics are gathered for the table as well as all the partitions (if any). If certain partition specs are specified, then statistics are gathered for only those partitions. When computing statistics across all partitions, the partition columns still need to be listed. As of [Hive 1.2.0](#), Hive fully supports qualified table name in this command. User can only compute the statistics for a table under current database if a non-qualified table name is used.

When the optional parameter NOSCAN is specified, the command won't scan files so that it's supposed to be fast. Instead of all statistics, it just gathers the following statistics:

- Number of files
- Physical size in bytes



### Version 0.10.0: FOR COLUMNS

As of [Hive 0.10.0](#), the optional parameter FOR COLUMNS computes column statistics for all columns in the specified table (and for all partitions if the table is partitioned). See [Column Statistics in Hive](#) for details.

To display these statistics, use DESCRIBE FORMATTED [db\_name.]table\_name column\_name [PARTITION (partition\_spec)].

## Examples

Suppose table Table1 has 4 partitions with the following specs:

- Partition1: (ds='2008-04-08', hr=11)
- Partition2: (ds='2008-04-08', hr=12)
- Partition3: (ds='2008-04-09', hr=11)
- Partition4: (ds='2008-04-09', hr=12)

and you issue the following command:

```
ANALYZE TABLE Table1 PARTITION(ds='2008-04-09', hr=11) COMPUTE STATISTICS;
```

then statistics are gathered for partition3 (ds='2008-04-09', hr=11) only.

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds='2008-04-09', hr=11) COMPUTE STATISTICS FOR COLUMNS;
```

then column statistics are gathered for all columns for partition3 (ds='2008-04-09', hr=11). This is available in Hive 0.10.0 and later.

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds='2008-04-09', hr) COMPUTE STATISTICS;
```

then statistics are gathered for partitions 3 and 4 only (hr=11 and hr=12).

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds='2008-04-09', hr) COMPUTE STATISTICS FOR COLUMNS;
```

then column statistics for all columns are gathered for partitions 3 and 4 only (Hive 0.10.0 and later).

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds, hr) COMPUTE STATISTICS;
```

then statistics are gathered for all four partitions.

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds, hr) COMPUTE STATISTICS FOR COLUMNS;
```

then column statistics for all columns are gathered for all four partitions (Hive 0.10.0 and later).

For a non-partitioned table, you can issue the command:

```
ANALYZE TABLE Table1 COMPUTE STATISTICS;
```

to gather statistics of the table.

For a non-partitioned table, you can issue the command:

```
ANALYZE TABLE Table1 COMPUTE STATISTICS FOR COLUMNS;
```

to gather column statistics of the table (Hive 0.10.0 and later).

If Table1 is a partitioned table, then for basic statistics you have to specify partition specifications like above in the analyze statement. Otherwise a semantic analyzer exception will be thrown.

However for column statistics, if no partition specification is given in the analyze statement, statistics for all partitions are computed.

You can view the stored statistics by issuing the [DESCRIBE](#) command. Statistics are stored in the Parameters array. Suppose you issue the analyze command for the whole table Table1, then issue the command:

```
DESCRIBE EXTENDED TABLE1;
```

then among the output, the following would be displayed:

```
... , parameters:{numPartitions=4, numFiles=16, numRows=2000, totalSize=16384, ...}, ....
```

If you issue the command:

```
DESCRIBE EXTENDED TABLE1 PARTITION(ds='2008-04-09', hr=11);
```

then among the output, the following would be displayed:

```
... , parameters:{numFiles=4, numRows=500, totalSize=4096, ...}, ....
```

If you issue the command:

```
desc formatted concurrent_delete_different partition(ds='tomorrow') name;
```

the output would look like this:


col_name		data_type		min	max	num_nulls	distinct_count	avg_col_len
max_col_len	num_trues	num_falses	bitvector	comment				
col_name	name	NULL	NULL	NULL	NULL	NULL	NULL	NULL
data_type	varchar(50)	NULL	NULL	NULL	NULL	NULL	NULL	NULL
min		NULL	NULL	NULL	NULL	NULL	NULL	NULL
max		NULL	NULL	NULL	NULL	NULL	NULL	NULL
num_nulls	0	NULL	NULL	NULL	NULL	NULL	NULL	NULL
distinct_count	2	NULL	NULL	NULL	NULL	NULL	NULL	NULL
avg_col_len	5.0	NULL	NULL	NULL	NULL	NULL	NULL	NULL
max_col_len	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL
num_trues		NULL	NULL	NULL	NULL	NULL	NULL	NULL
num_falses		NULL	NULL	NULL	NULL	NULL	NULL	NULL
bitVector		NULL	NULL	NULL	NULL	NULL	NULL	NULL
comment	from deserializer	NULL	NULL	NULL	NULL	NULL	NULL	NULL

If you issue the command:

```
ANALYZE TABLE Table1 PARTITION(ds='2008-04-09', hr) COMPUTE STATISTICS NOSCAN;
```

then statistics, number of files and physical size in bytes are gathered for partitions 3 and 4 only.

ANALYZE TABLE <table1> CACHE METADATA

**Feature not implemented**

Hive Metastore on HBase was discontinued and removed in Hive 3.0.0. See [HBaseMetastoreDevelopmentGuide](#)

When Hive metastore is configured to use HBase, this command explicitly caches file metadata in HBase metastore.

*The goal of this feature is to cache file metadata (e.g. ORC file footers) to avoid reading lots of files from HDFS at split generation time, as well as potentially cache some information about splits (e.g. grouping based on location that would be good for some short time) to further speed up the generation and achieve better cache locality with consistent splits.*

```
ANALYZE TABLE Table1 CACHE METADATA;
```

See feature details in [HBase Metastore Split Cache](#) and [\(HIVE-12075\)](#)

Current Status (JIRA)

type	key	summary	assignee	reporter	priority	status	resolution	created	updated	due
------	-----	---------	----------	----------	----------	--------	------------	---------	---------	-----



Can't show details. Ask your admin to add this Jira URL to the allowlist.

[View these issues in Jira](#)

---