



Hive Good Practices

These are global best practices, they apply to any kind of project.

Authentication

HiveServer2 can be configured to authenticate all connections; by default, it allows any client to connect. HiveServer2 supports either Kerberos or LDAP authentication; configure this in the `hive.server2.authentication` property in the `hive-site.xml` file. You can also configure Pluggable Authentication, which allows you to use a custom authentication provider for HiveServer2; and HiveServer2 Impersonation, which allows users to execute queries and access HDFS files as the connected user rather than the super user who started the HiveServer2 daemon. For more information, see Hive Security Configuration.

ClientPort - `hive.zookeeper.client.port`

If ZooKeeper is not using the default value for ClientPort, you need to set `hive.zookeeper.client.port` in `/etc/hive/conf/hive-site.xml` to the same value that ZooKeeper is using. Check `/etc/zookeeper/conf/zoo.cfg` to find the value for ClientPort. If ClientPort is set to any value other than 2181 (the default), set `hive.zookeeper.client.port` to the same value. For example, if ClientPort is set to 2222, set `hive.zookeeper.client.port` to 2222 as well.

Column Delimiters

Hive wasn't escaping characters such as “,” (comma) by default, so changed CSVSerde with single quote (') enclosed fields and back-slash (\) escaping.

After, I've enabled escaping, quotation in Sqoop export, so now in SQL server we have correct values.

Compression

Keeping data compressed in Hive tables has, in some cases, been known to give better performance than uncompressed storage; both in terms of disk usage and query performance.

You can import text files compressed with Gzip or Bzip2 directly into a table stored as `TextFile`. The compression will be detected automatically and the file will be decompressed on-the-fly during query execution. For example:

```
CREATE TABLE raw (line STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INPATH '/tmp/weblogs/20090603-access.log.gz' INTO TABLE raw;
```

The table 'raw' is stored as a `TextFile`, which is the default storage. However, in this case Hadoop will not be able to split your file into chunks/blocks and run multiple maps in parallel. This can cause underutilization of your cluster's 'mapping' power.

The recommended practice is to insert data into another table, which is stored as a `SequenceFile`. A `SequenceFile` can be split by Hadoop and distributed across map jobs whereas a GZIP file cannot be.

The value for `io.seqfile.compression.type` determines how the compression is performed. Record compresses each value individually while `BLOCK` buffers up 1MB (default) before doing compression.

Selection of compression format will be influenced by how the data will be used. For archival purposes you may choose the most compact compression available, but if the data will be used in processing jobs such as MapReduce, you'll likely want to select a splittable format. Splittable formats provide the ability for Hadoop to split files into chunks for processing, which is critical to efficient parallel processing. We'll discuss compression types and considerations, including the concept of splittability, later in this chapter.

Note also that in many, if not most cases, the use of a container format such as `SequenceFiles` or `Avro` will provide advantages which makes it a preferred format for most file types, including text — among other things these container formats provide functionality to support splittable compression. We'll also be covering these container formats later in this chapter.

Snappy is a compression library developed at Google, and, like many technologies that come from Google, Snappy was designed to be fast. The trade off is that the compression ratio is not as high as other compression libraries. One thing to note is that Snappy is intended to be used with a container format, like `Sequence Files` or `Avro Data Files`, rather than being used directly on plain text, for example, since the latter is not splittable and can't be processed in parallel using MapReduce.

Container

The use of a container format such as `Sequence Files` or `Avro` will provide advantages which makes it a preferred format for most file types, including text — among other things these container formats provide functionality to support splittable compression.

Metadata - Check Table Layout and Information

```
DESCRIBE FORMATTED <table_name> - Hive and Impala
```

Metadata - Invalidates metadata. All table metadata will be reloaded on the next access

```
invalidate metadata <table_name>; - Hive and Impala
```

Metadata - Refreshes the metadata immediately. It is a faster, incremental refresh

```
refresh <table_name>; - Hive and Impala
```

Distinct

This will cause Hive to use 1 reducer, which is a huge performance loss:

```
Select count(DISTINCT user_id) From
```

```
Select count(1) From (Select distinct user_id From ) u
```

SequenceFiles are well supported within the Hadoop ecosystem, however their support outside of the ecosystem is limited. A common use case for SequenceFiles is as a container for smaller files. Since Hadoop is optimized for large files, packing smaller files into a SequenceFile makes the storage and processing of these files much more efficient.

The goal of Hive indexing is to improve the speed of query lookup on certain columns of a table. Without an index, queries with predicates like 'WHERE tab1.col1 = 10' load the entire table or partition and process all the rows. But if an index exists for col1, then only a portion of the file needs to be loaded and processed. The improvement in query speed that an index can provide comes at the cost of additional processing to create the index and disk space to store the index.

Querying

```
SELECT pkg_name, count(distinct imei) FROM mobile_kill_appinfo
where pkg_name in ('appinventor.ai.funayamajogos.Caixa',
'appinventor.ai.funayamajogos.BancodoBrasil')
and concat(year, '-', month, '-', day) >= '2014-10-31' and
concat(year, '-', month, '-', day) <= '2014-11-18'
group by pkg_name
```

Mapping column family to Hive's map doesn't select cells with empty values

Query Editors

Data Browsers

Workflows

Search

HBase Browser

Начало - HBase / my_total_android_user_activity

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix

user1

a: 2014-11-25	a: 2014-12-02	
v	v	

user2

a: 2014-11-10	a: 2014-12-02	
v	v	

user_id	activity
user1	{"2014-12-02": "v"}
user2	{"2014-11-10": "v", "2014-12-02": "v"}

<https://issues.apache.org/jira/browse/HIVE-2987> <https://issues.apache.org/jira/browse/HIVE-2987>

select * from total_android_user_profile works well. It returns 2 records:

EditorMy QueriesSaved QueriesИстория

1 select * from total_android_user_profile

ExecuteSave as...Explainor create aNew query

...

Recent queriesQueryLogColumnsResultsChart

total_android_user_profile.user_idtotal_android_user_profile.versiontotal_android_user_profile.channeltotal_android_user_profile.first_inst

0	user1	20	100	NULL
1	user2	30	200	NULL

As well as select user_id, version, first_install from total_android_user_profile:

EditorMy QueriesSaved QueriesИстория

1 select user_id, version, first_install from total_android_user_profile

ExecuteSave as...Explainor create aNew query

...

Recent queriesQueryLogColumnsResultsChart

user_idversionfirst_install

0	user1	20	NULL
1	user2	30	NULL

But select user_id, first_install from total_android_user_profile doesn't work properly. It returns empty result:

Editor

My Queries

Saved Queries

История

1

```
select user_id, first_install from total_android_user_profile
```

Execute

Save as...

Explain

or create a

New query

...

Recent queries

Query

Log

Columns

Results

Chart

The operation has no results.

It happens because of first_install is null in my database and only null-values (except for the keys are selected).

One more problem with null-values

<https://issues.apache.org/jira/browse/HIVE-7566> [<https://issues.apache.org/jira/browse/HIVE-7566>]

select count(*) as cnt from total_android_user_profile – 2 records as expected:

Editor

My Queries

Saved Queries

История

1

```
select count(*) as cnt from total_android_user_profile
```

Execute

Save as...

Explain

or create a

New query

...

Recent queries

Query

Log

Columns

Results

Chart

cnt

0

cnt

2

But select count(*) as cnt from total_android_user_profile where first_install is null returns zero count:

EditorMy QueriesSaved QueriesИстория

1 select count(*) as cnt from total_android_user_profile where first_install is null

Execute

Save as...

Explain

or create a

New query

Recent queriesQueryLogColumnsResultsChart

cnt

0

As well as `select count(*) as cnt from total_android_user_profile where first_install is NOT null`. I just replaced “is null” with “is not null”. So first_install doesn’t meet both these conditions:

EditorMy QueriesSaved QueriesИстория

1 select count(*) as cnt from total_android_user_profile where first_install is NOT null

Execute

Save as...

Explain

or create a

New query

Recent queriesQueryLogColumnsResultsChart

cnt

0

Table Lock Manager (Required)

You must properly configure and enable Hive's Table Lock Manager. This requires installing ZooKeeper and setting up a ZooKeeper ensemble