LanguageManual Sampling

- Sampling Syntax
  - Sampling Bucketized Table
  - Block Sampling

Sampling Syntax

Sampling Bucketized Table

```
table_sample: TABLESAMPLE (BUCKET x OUT OF y [ON colname])
```

The TABLESAMPLE clause allows the users to write queries for samples of the data instead of the whole table. The TABLESAMPLE clause can be added to any table in the FROM clause. The buckets are numbered starting from 1. **colname** indicates the column on which to sample each row in the table. colname can be one of the non-partition columns in the table or **rand()** indicating sampling on the entire row instead of an individual column. The rows of the table are 'bucketed' on the colname randomly into y buckets numbered 1 through y. Rows which belong to bucket x are returned.

In the following example the 3rd bucket out of the 32 buckets of the table source. 's' is the table alias.

```
SELECT *
FROM source TABLESAMPLE(BUCKET 3 OUT OF 32 ON rand()) s;
```

Input pruning: Typically, TABLESAMPLE will scan the entire table and fetch the sample. But, that is not very efficient. Instead, the table can be created with a CLUSTERED BY clause which indicates the set of columns on which the table is hash-partitioned/clustered on. If the columns specified in the TABLESAMPLE clause match the columns in the CLUSTERED BY clause, TABLESAMPLE scans only the required hash-partitions of the table.

Example:

So in the above example, if table 'source' was created with 'CLUSTERED BY id INTO 32 BUCKETS'

```
    TABLESAMPLE(BUCKET 3 OUT OF 16 ON id)
```

would pick out the 3rd and 19th clusters as each bucket would be composed of (32/16)=2 clusters.

On the other hand the tablesample clause

```
    TABLESAMPLE(BUCKET 3 OUT OF 64 ON id)
```

would pick out half of the 3rd cluster as each bucket would be composed of (32/64)=1/2 of a cluster.

For information about creating bucketed tables with the CLUSTERED BY clause, see Create Table (especially Bucketed Sorted Tables) and Bucketed Tables.

Block Sampling

Block sampling is available starting with Hive 0.8. Addressed under JIRA - https://issues.apache.org/jira/browse/HIVE-2121

```
block_sample: TABLESAMPLE (n PERCENT)
```

This will allow Hive to pick up at least n% data size (notice it doesn't necessarily mean number of rows) as inputs. Only CombineHiveInputFormat is supported and some special compression formats are not handled. If we fail to sample it, the input of MapReduce job will be the whole table/partition. We do it in HDFS block level so that the sampling granularity is block size. For example, if block size is 256MB, even if n% of input size is only 100MB, you get 256MB of data.

In the following example the input size 0.1% or more will be used for the query.

```
SELECT *
FROM source TABLESAMPLE(0.1 PERCENT) s;
```

Sometimes you want to sample the same data with different blocks, you can change this seed number:

```
set hive.sample.seednumber=<INTEGER>;
```

Or user can specify total length to be read, but it has same limitation with PERCENT sampling. (As of Hive 0.10.0 - https://issues.apache.org/jira/browse/HIVE-3401)

```
block_sample: TABLESAMPLE (ByteLengthLiteral)

ByteLengthLiteral : (Digit)+ ('b' | 'B' | 'k' | 'K' | 'm' | 'M' | 'g' | 'G')
```

In the following example the input size 100M or more will be used for the query.

```
SELECT *
FROM source TABLESAMPLE(100M) s;
```

Hive also supports limiting input by row count basis, but it acts differently with above two. First, it does not need CombineHiveInputFormat which means this can be used with non-native tables. Second, the row count given by user is applied to each split. So total row count can be vary by number of input splits. (As of Hive 0.10.0 - https://issues.apache.org/jira/browse/HIVE-3401)

```
block_sample: TABLESAMPLE (n ROWS)
```

For example, the following query will take the first 10 rows from each input split.

```
SELECT * FROM source TABLESAMPLE(10 ROWS);
```