BINFOR-105: Cloud Computing (Winter Term 2019/20)
Dr. Vinu Venugopal
Exercise Sheet #2 – Demo on Nov. 12, 2019

- *All solutions may be submitted in groups of up to 3 students.*
- *Send your solutions before the due date per email to* `cloudcomputing@uni. lu` *in a single zip file (containing just the java files or scripts) using your group id as the file name.*
- *Include some brief instructions on how to run your solution to each problem in a file called* `problem_X.txt`.
- *All group members are required to be present in the class for the demonstration of the solutions on the due date.*

II. Pig Latin & Hive

For this exercise, please consider downloading the Yago dataset available on the course homepage on `moodle.uni.lu`.

Processing YAGO dataset **6 Points**

Problem 1. Find the top three frequently occurring *predicates* in the Yago dataset using:

(i) operators available in the Pig Latin scripting language;
3 Points

(ii) operators available in HiveQL.
3 Points

Grouping and Joining **8 Points**

Problem 2. Identify all the given-names (corresponding to `hasGivenName` predicate) of persons who are associated with more than one `livesIn` predicates from the Yago dataset using:

(i) the relational operations (joins, grouping, etc.) available in the Pig Latin scripting language;
4 Points

(ii) the relational operations which are available in HiveQL.
4 Points

Data Clustering & Bucketized Merge-Join **22 Points**

Problem 3. Write a HiveQL query to find all the subjects (`x`) and objects (`y` and `z`) matching the pattern: `?x <hasGivenName> ?y.  ?x <livesIn> ?z.`, from the Yago dataset.

- Implement this problem by:
  (i) by considering *partitioning* and *bucketing*;
  (ii) by considering *partitioning* but not *bucketing*;
  (iii) by considering neither *partitioning* nor *bucketing*.
- For the first case alone perform a *Bucketized Merge-Join* by enabling the necessary parameters (see the note below).

- Compare the run time of the three cases by performing your experiments on an AWS EC2 instance.
- For case (i):
  - Load the entire triples from the given `yago.tsv` files into a table named `yago` having three columns: *subject, predicate* and *object.*
  - Create a new table, named `yago_part_buck`, with a partition based on the predicate column and clustered based on the subject column.
  - Load data (statically) into the partitions for all the 29 predicates[1] in the dataset. This loading could be done by inserting data into the partitioned table from the `yago` table specifying the partition key – you may write all the insert statements into a single HiveQL script for loading data.
  - Write a HiveQL query to find the required pattern from the `yago_part_buck` table.

Note: You can set the following hive parameters to true (as given below), to enable the bucketized merge-join.

```
set hive.auto.convert.sortmerge.join=true;
set hive.optimize.bucketmapjoin = true;
set hive.optimize.bucketmapjoin.sortedmerge = true;
```

[1] `<actedIn>, <hasAcademicAdvisor>, <hasChild>, <hasFamilyName>, <hasWebsite>, <hasWonPrize>, <isInterestedIn>, <isKnownFor>, <directed>, <edited>, <graduatedFrom>, <hasGender>, <hasMusicalRole>, <isCitizenOf>, <isMarriedTo>, <isPoliticianOf>, <playsFor>, <worksAt>, <wroteMusicFor>, <created>, <diedIn>, <hasGivenName>, <influences>, <isAffiliatedTo>, <isLeaderOf>, <livesIn>, <owns>, <participatedIn>, <wasBornIn>`