Big Data Management (MIRI/MDS/BDMA)

Final Exam

Family name:_____ Given name: _____

1. (20%) Compare a B-tree and an LSM-tree in the context of the **RUM conjecture** (i.e., as an answer to this question, three brief explanations of the form "From the perspective of X, Y-tree is better than Z-tree, because of this and that." are expected).

   (a) R.....

   .....
   .....
   .....

   (b) U.....

   .....
   .....
   .....

   (c) M.....

   .....
   .....
   .....

2. (20%) Given a file with $3.2GB$ of raw data stored in an HDFS cluster of 50 machines, and containing $16 \cdot 10^5$ rows in a **Parquet file**; consider you have a query over an attribute "$A =$ constant" and this attribute contains only 100 different and equiprobable values. Assuming any kind of compression has been disabled, explicit any assumption you need to make and give the amount of raw data (i.e., do not count metadata) it would need to fetch from disk.

   - Replication factor: 3 (default)
   - Chunk size: $128MB$ (default)
   - Rowgroup size: $32MB$

   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....
   .....

3. (20%) Given an empty **Consistent Hash** with $h(x) = x\%32$ (i.e., we directly take module 32 to both the keys and the bucket IDs), and unlimited capacity in each bucket, consider you have a cluster of four machines with IDs 19, 22, 75, 92, and draw the result of inserting the following keys in the given order: 12, 4, 10, 49, 42, 60, 63, 53, 47, 27, 26, 28, 13, 52.

4. (20%) Assume you have a MongoDB collection which occupies 6 chunks **UNevenly distributed** in 3 shards (i.e., 1, 2 and 3 chunks per shard respectively). Being the document Id also the shard key, the chunk of a document is determined **by means of a hash function**. Assuming that accessing one document takes one time unit (existing indexes are used at no cost) and we have 6,000 documents in the collection, k of which have value "YYY" for attribute "other", how many time units would take the following operations[1]:

   (a) $FindOne(\{\_id : "XXX"\})$

   (b) $Find(\{\_id : \{\$in : [1, .., 3000]\}\})$, being [1,..,6000] the range of existing IDs.

   (c) $Find(\{other : "YYY"\})$, being the attribute indexed.

   (d) $Find(\{other : "YYY"\})$, being the attribute NOT indexed.

[1] As typically in RDBMS optimizers, assume uniform distribution of values and statistical independence between pairs of attributes.

Family name:_____ Given name: _____

5. (20%) Given two files containing the following kinds of data:

Employees.txt with fields: EmployeeID; EmployeeName; YearlySalary; CityOfResidence; SiteOfWork

```
EMP4;RICARDO;250000;MADRID;DPT4
EMP5;EULALIA;150000;BARCELONA;DPT5
EMP6;MIQUEL;125000;BADALONA;DPT5
EMP7;MARIA;175000;MADRID;DPT6
EMP8;ESTEBAN;150000;MADRID;DPT6
...
```

Departments.txt with fields: SiteID; DepartmentName; StreetNumber; StreetName; City

```
DPT1;DIRECCIO;10;PAU CLARIS;BARCELONA
DPT2;DIRECCIO;8;RIOS ROSAS;MADRID
DPT3;MARKETING;1;PAU CLARIS;BARCELONA
DPT4;MARKETING;3;RIOS ROSAS;MADRID
...
```

Consider the following **PySpark code** and answer the questions bellow.

```
1: A = sc.textFile("employees.txt").map(lambda l: tuple(l.split(";")))
2: B = sc.textFile("departments.txt").map(lambda l: tuple(l.split(";")))
3: C = A.map(lambda t: t[3])
4: D = B.map(lambda t: t[1])
5: E = D.cartesian(C)
6: F = B.map(lambda t: (t[1],t[4]))
7: G = E.subtract(F)
8: H = G.map(lambda t: t[0])
9: R = D.subtract(H)
```

(a) State in natural language the corresponding query it would answer?

.....

.....

.....

.....

(b) Clearly indicate any mistake or improvement you can fix/make in the code? For each of them give (1) the line number, (2) pseudo-code to implement the fix, and (3) brief rationale.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....