Q1 Directions
6 Points

For this midterm you are allowed the use of your two-page cheat sheet, but no resources on the Internet. It is an 80-minute exam.

Clarification questions can be posted **privately only** to Piazza, and will be responded to there.

Before you submit your exam, please consult the Exam Clarifications here: https://docs.google.com/document/d/1lBhXtt5_0jd1g7eg7wKyGHrz20Blk-5AXMtuqSBryx0/edit?usp=sharing.

- [x] I've read this and understand the rules.

Q2 Capturing data
3 Points

Which can capture more information overall:

- ○ relations
- ○ graphs
- ○ JSON / hierarchical data
- ◉ they are equivalent

Q3 Query operations
4 Points

Which of the following are true (select all that apply):

- [x] the results of an inner join are a subset of the results of an outer join
- [ ] the main operator for combining different rows from similar tables is the outer join
- [ ] the *applymap* operator is used to combine tables
- [x] a filter or select operation can be performed in parallel across each tuple

Q4 Knowledge representation

3 Points

If we are given a conceptual class of entities C and a subclass of C called S, we know that:

- (•) every member of class S must also be a member of class C
- ( ) class C represents a subset of class S
- ( ) there is no formal relationship between S and C
- ( ) class C has every property in class S

Q5 NoSQL

3 Points

Choose the *best* answer from the following.

- ( ) NoSQL databases cannot store nested objects.
- (•) NoSQL databases typically have limited notions of consistency

- [x] NoSQL databases typically have limited notions of consistency.
- [ ] SQL databases are non-relational.
- [ ] SQL databases are key-value based.

Q6 Integration
3 Points

For the strings "danger" and "roger," how many 3-grams will be in common?

- [ ] 1
- [ ] 2
- [x] 3
- [ ] 4
- [ ] 6

Q7 Computer architecture
3 Points

Given a list of 768 single-byte characters, an iteration through the characters in the array, and a cache size of 256B, we can amortize memory accesses by up to a factor of:

- [ ] 3
- [ ] 16
- [x] 256
- [ ] 768

Q8 Sharding
4 Points

Give the **best** answer here. Given a table $R(A, B, C)$ with 10M entries, if we use hash-based sharding on attribute A in this table, we can support up to how many worker nodes running in parallel:

- ○ 1,000
- ○ 1M
- ○ 10M
- ◉ number of unique values of A
- ○ number of unique tuples in R

Q9 Queries
3 Points

Suppose we have a database of students, courses, and enrollments. If we want to know the number of *unique last names* of students who are enrolled in courses, we can:

- ○ left outerjoin all students with enrollments, count the number of (distinct) student last names
- ◉ inner join all students with enrollments, count the number of (distinct) student last names
- ○ union all students with enrollments, count the number of (distinct) student last names
- ○ this query is not possible to express in SQL, Pandas, or relational operators

Q10 Statistical tests

3 Points

Suppose we apply a statistical test, it indicates that the probability of the null hypothesis is under our α level, and we decide that our candidate hypothesis holds. If, in reality our candidate hypothesis is wrong, then:

- (●) we have a false positive
- ( ) we have a false negative
- ( ) we used the wrong test for the distribution
- ( ) we need to use the Bonferroni correction

Q11 Indexing

4 Points

Indexing a relation or dataframe $R(A, B, C)$ by attribute A helps with (check all that apply):

- [x] joining on attribute A
- [x] filtering on attribute A
- [x] projecting attribute A
- [x] grouping by attribute A
- [ ] projecting attributes (A, B)

Q12 Breadth-first search

3 Points

Which does less overall work, for most real-world graphs:

- (selected) sequential (centralized) breadth-first search
- parallel breadth-first search
- they do exactly the same amount of work
- breadth-first search cannot be parallelized

Q13 Partitioning data

3 Points

Which of the following are likely to make it *more difficult* to balance our workload across machines, in a sharded query processing setting:

- dataframes are of different sizes from one another
- certain values occur frequently, within an attribute used in selection operations
- (selected) certain values occur frequently, within a join key
- all values within a grouping key appear exactly once

Q14 Link analysis

3 Points

In the PageRank algorithm with the decay factor added, the total PageRank in the system should, across iterations:

- increase
- decrease
- (selected) remain constant

- [ ] any of the above, it depends on the structure of the graph

Q15 Matrices for machine learning

3 Points

When we convert from a dataframe R with k attributes and n rows to a matrix M suitable for machine learning, how many columns can the matrix be? Consider cases where the dataframe is not purely numeric.

- [ ] k
- [x] $k \cdot n$
- [ ] n
- [ ] n^2

Q16 Distributed processing

6 Points

Name two relational operations in Spark (dataframes or SQL) require the system to shard on a particular key.

Q16.1 Relational operator 1

3 Points

Your first answer:

GROUP BY

Q16.2 Relational operator 2
3 Points

Your second answer:

JOIN

Q17 Speedups
8 Points

Name two types of **data structures** we (Pandas, Spark, or the programmer) can take advantage of to speed up dataframe or query processing.

Q17.1 Data structure 1
4 Points

Your first answer:

Dictionary

Q17.2 Data structure 2
4 Points

Your second answer:

B+ Tree

Q18 Long-form query
11 Points

Suppose we have a dataset about classes, and we are curious how often we have multiple students **with the same first name** enrolled in a class. The tables are $Students(id, first, last)$ and $Enrolled(studentid, courseid)$.

The data is available in Spark dataframes. The $Students$ table (dataframe) is sharded by id and the $Enrolled$ table is not sharded by any particular key (i.e., tuples are randomly partitioned aross our worker nodes).

Q18.1 The query

6 Points

Write a query to find all such results (either name, class, count; or name, count), using Spark or Pandas operations or SQL, or pseudocode that gives enough detail to understand which operators are being performed.

ans = spark.sql('''SELECT Students.first, Enrolled.courseid, count(*)
FROM Students JOIN Enrolled on Students.id = Enrolled.studentid
GROUP BY Students.first, Enrolled.courseid''')

Q18.2 Distributed execution

5 Points

Given your query, indicate where Spark would have to *repartition* or shard (also called *shuffle* or *exchange*) the data to perform the computation. Specify what the repartition key would be.

Shard on Enrolled.studentid, and on both group by columns: Students.first and Enrolled.courseid.

Q19 Costs

11 Points

Suppose we are given JSON document on people that looks like the following fragment.

```
[
  {"id": 1,
   "name": "Jun",
   "parent_ids": [],
   "child_ids": [4, 5]
  },
  {"id": 2,
   "name": "Maya",
   "parent_ids": [],
   "child_ids": [4, 5]
  },
   ...
  {"id": 4,
   "name": "Ava",
   "parent_ids": [1, 2],
   "child_ids": []
  },
]
```

Q19.1 Schema for hierarchical data

5 Points

Propose a **schema** (with attribute types) for the above data. Use the syntax:

Table(attrib1: type1, attrib2: type2, ...)

and specify what would be the *keys* and *foreign keys* or references.

Person(id: int, name:string)

HasParent(parent_id: int, child_id: int)

Keys: id of person table
Foreign keys: parent_id of HasParent, child_id of HasParent

Q19.2 Populating tables from JSON

6 Points

Using Pandas, Spark SQL, or the equivalent -- show how we would read the JSON data and populate the table(s).

```
schema = StructType([
StructField("id", IntegereType()),
StructField("name", StringType()),
StructField("parent_ids", ArrayTypee()),
StructField("child_ids",ArrayTypee())])

raw_data = spark.read.json("data.json", schema=schema)
raw_data.creeateeOrReplaceTempView("data")
person = spark.sql('''SELECT id, name FROM data''')
hasParent = spark.sql('''SELECT id as parent_id, col as child_id
FROM
(SELECT id, POSEXPLODE(child_ids)  FROM data)
WHERE col IS NOT NULL '''
```

Midterm 1

GRADED

STUDENT

Claudia Jiyun Zhu

TOTAL POINTS

85.5 / 87 pts

QUESTION 1

Directions **6** / 6 pts

QUESTION 2

Capturing data **3** / 3 pts

QUESTION 3

Query operations **4** / 4 pts

QUESTION 4

Knowledge representation **3** / 3 pts

QUESTION 5

NoSQL **3** / 3 pts

QUESTION 6

Integration **3** / 3 pts

QUESTION 7

Computer architecture **3** / 3 pts

QUESTION 8

Sharding **4** / 4 pts

QUESTION 9

Queries **3** / 3 pts

QUESTION 10

Statistical tests **3** / 3 pts

QUESTION 11

Indexing **4** / 4 pts

QUESTION 12

Breadth-first search **3** / 3 pts

QUESTION 13

Partitioning data **3** / 3 pts

QUESTION 14

Link analysis **3** / 3 pts

QUESTION 15

Matrices for machine learning **3** / 3 pts

QUESTION 16

Distributed processing **6** / 6 pts

16.1 Relational operator 1 **3** / 3 pts

16.2 Relational operator 2 **3** / 3 pts

QUESTION 17

Speedups **8** / 8 pts

17.1 Data structure 1 **4** / 4 pts

17.2 Data structure 2 **4** / 4 pts

QUESTION 18

Long-form query **9.5** / 11 pts

18.1 The query **4.5** / 6 pts

18.2 Distributed execution **5** / 5 pts

QUESTION 19

Costs **11** / 11 pts

19.1 Schema for hierarchical data **5** / 5 pts

19.2 Populating tables from JSON **6** / 6 pts