Techniques for Large-Scale Data: Exam

University of Gothenburg | Chalmers University of Technology
Department of Computer Science and Engineering
Period 4, 2020 (DIT873 (and re-exam in DIT871/872) / DAT346)
Instructors: Dr. Alexander Schliep (Examiner) with Dr. Graham Kemp

Information:

- The exam takes place from 8:30–12:30 on Wednesday, June 3, 2020.
- The instructors or a representative will be available via Zoom throughout the exam.
- Exam review: See the Canvas course website for details.
- You can earn are a total of 45 points in 7 questions in the exam.
- Additionally, Zoom poll percentage points will contribute to the exam points as bonus points; the maximum will be 4.5 (or 10% of the total exam points) added for a 100% Zoom poll result.
- Grades for GU students (DIT873) are normally determined as follows: $\geq$ 70 % for grade VG; $\geq$ 40 % for grade G. Grades for Chalmers students (DAT346) are normally determined as follows: $\geq$ 80 % for grade 5; $\geq$ 60 % for grade 4; $\geq$ 40 % for grade 3.

Instructions:

- All aids are permitted.
- Please follow the exam instructions on Canvas (and instructions provided by Chalmers).
- Upload your solution as one PDF to Canvas.
- Begin the answer to each question on a new page. Write page number and question number on **every** page.
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions.
- Indicate clearly if you make any assumptions that are not given explicitly in the question.
- Show **ALL** your work. You will get little or no credit for an unexplained answer. Please indicate why a specific computation or transformation is appropriate. The points of each question appear in parentheses; use this for guiding your time.
- There is no need to compute numerical answers; you may leave Binomials, factorials and fractions, should they arise, as is.

Question 1 [*4 points total*]

Suppose we have relations $R(a,b)$, $S(c,d)$ and $T(e,f)$ and the query:

```
SELECT a, c, e
FROM   R, S, T
WHERE  b=d AND b=f
```

Draw *two* alternative logical query plans for this query:

(a) the first tree should contain two join operations;

(b) the second tree should contain one join operation and one Cartesian product operation.

Discuss the efficiency of these alternative query plans.

Question 2 [*4 points total*]

Suppose that each node in a B^+-tree can have up to *three* pointers.
Draw a B^+-tree that contains the data values 1, 3, 5, 7, 9, 11, 13, 15, 17 and 19.

Question 3 [*8 points total*]

(a) [*4 pts*] A SPARQL query that accesses DBpedia is translated into the following SQL query. What was the original SPARQL query?

```
SELECT __id2in ( "s_12_3_t0"."S") AS "x",
  __ro2sq ( "s_12_3_t2"."O") AS "y"
FROM DB.DBA.RDF_QUAD AS "s_12_3_t0"
  INNER JOIN DB.DBA.RDF_QUAD AS "s_12_3_t1"
  ON (
    "s_12_3_t0"."S" = "s_12_3_t1"."S")
  INNER JOIN DB.DBA.RDF_QUAD AS "s_12_3_t2"
  ON (
    "s_12_3_t0"."S" = "s_12_3_t2"."S"
    AND
    "s_12_3_t1"."S" = "s_12_3_t2"."S")
WHERE
  "s_12_3_t0"."G" = __i2idn ( __bft( 'http://dbpedia.org' , 1))
  AND
  "s_12_3_t0"."P" = __i2idn ( __bft( 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type' , 1))
  AND
  "s_12_3_t0"."O" = __i2idn ( __bft( 'http://dbpedia.org/ontology/University' , 1))
  AND
  "s_12_3_t1"."G" = __i2idn ( __bft( 'http://dbpedia.org' , 1))
  AND
  "s_12_3_t1"."P" = __i2idn ( __bft( 'http://dbpedia.org/ontology/country' , 1))
  AND
  "s_12_3_t1"."O" = __i2idn ( __bft( 'http://dbpedia.org/resource/United_Kingdom' , 1))
  AND
  "s_12_3_t2"."G" = __i2idn ( __bft( 'http://dbpedia.org' , 1))
  AND
  "s_12_3_t2"."P" = __i2idn ( __bft( 'http://dbpedia.org/property/established' , 1))
  AND
 ( "s_12_3_t2"."O" >  1400)
  AND
 ( "s_12_3_t2"."O" <  1800)
```

(b) [*4 pts*] Write a SPARQL query that accesses WikiData to answer the same question as the query in part (a). (Hint: The built-in function YEAR() can be used to extract the year from a date.)

Question 4 [*8 points total*]
On pages 5–7 of this document you find an OWL ontology in turtle format.
All parts of this question refer to that ontology.

(a) [*4 pts*] Draw an Entity-Relationship diagram (E-R diagram) that shows the classes, relationships and attributes that are defined in the ontology.

(b) [*1 pts*] Give OWL statements that declare inverses of object properties takesPartIn and isStageOf.

(c) [*3 pts*] Suppose that a database based on the ontology file is implemented in Neo4j. Give a Cypher query that finds the the name of the rider who wins a stage that finishes in Paris.

Question 5 [*4 points total*]
A pizza company uses AI to check the quality of its products:

> DOM Pizza Checker uses advanced machine learning, artificial intelligence and sensor technology to identify pizza type, even topping distribution and correct toppings.

(Source: `https://dompizzachecker.dominos.com.au/`)

Discuss whether deploying the DOM Pizza Checker raises potential ethical issues.

Question 6 [*15 points total*]
Assume you need to analyze 27 TB of historical log-file information from a large telecommunication equipment manufacturer. The log files are text files; each row of comma separated values has the same structure,

```
Error class, Error code, Log Message.
```

Here error class indicates one of four subsystems—memory, CPU, network, power encoded as strings MEMERR, CPUERR, NETERR, PWRERR, respectively—in which the error occurred, error type is one of about 26000 numerical 5-digit error codes (in the range 10000,...,37000), and log message is a natural language text giving further details.

The data is made available on a cluster consisting of twenty servers with 64 cores each and local disks, where the 27TB of data is stored in a distributed manner in the Hadoop file system (HDFS).

Solve the problem of reporting the 50 most-frequent error codes in each error class and, additionally, the 10 most-frequent error codes irrespective of error class. Your approach should be efficient. That is use parallel computations as much as possible, and avoid unnecessary communication.

Sketch your solution either graphically, with bullet points, or commented pseudo-code (or Python code) and explain how the communication between parallel tasks is handled. You do not have to explain File I/O. Do explain the data flow and how the execution progresses to the final result. Be specific, so that someone with familiarity with Python, MapReduce and Spark would have enough details to implement your solution. In particular, specify the keys you propose to use in MapReduce or Spark *explicitly*.

(a) [*6 pts*] Propose a solution using the MapReduce-framework. Please give your best-case estimate (that is, ignoring pathological inputs such as 27TB of identical lines in the log) on how many cores/servers each specific step of your MapReduce solution can be executed in parallel.

(b) [*5 pts*] Propose a solution using Spark. Again, give your best-case estimate on how many cores/servers each specific step of your solution can be executed in parallel.

(c) [*4 pts*] Discuss the differences and argue whether the Spark model of parallel programming has an advantage over MapReduce for this problem. Please be specific.

Question 7 [*2 points total*]

As a junior data scientist recently hired to Reynholm Industries you are tasked with overhauling and redesigning the IT infrastructure and accelerating existing computational workloads. A lot of the existing analytics code is utilizing one core. Outline your decision process, including key variables, when deciding whether a specific program can and should be parallelized.

```
@prefix : <http://www.semanticweb.org/kemp/ontologies/2020/5/untitled-ontology-23#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.semanticweb.org/2020-06-03-exam> .

<http://www.semanticweb.org/2020-06-03-exam> rdf:type owl:Ontology .

#################################################################
#    Object Properties
#################################################################

###  http://www.semanticweb.org/2020-06-03-exam#finishesAt
:finishesAt rdf:type owl:ObjectProperty ;
            rdfs:domain :Stage ;
            rdfs:range :City .


###  http://www.semanticweb.org/2020-06-03-exam#includesSprint
:includesSprint rdf:type owl:ObjectProperty ;
                rdfs:domain :Stage ;
                rdfs:range :IntermediateSprint .


###  http://www.semanticweb.org/2020-06-03-exam#isStageOf
:isStageOf rdf:type owl:ObjectProperty ;
           rdfs:domain :Stage ;
           rdfs:range :Race .


###  http://www.semanticweb.org/2020-06-03-exam#raceWinner
:raceWinner rdf:type owl:ObjectProperty ;
            rdfs:domain :Race ;
            rdfs:range :Rider .


###  http://www.semanticweb.org/2020-06-03-exam#sprintWinner
:sprintWinner rdf:type owl:ObjectProperty ;
              rdfs:domain :IntermediateSprint ;
              rdfs:range :Rider .


###  http://www.semanticweb.org/2020-06-03-exam#stageWinner
:stageWinner rdf:type owl:ObjectProperty ;
             rdfs:domain :Stage ;
             rdfs:range :Rider .


###  http://www.semanticweb.org/2020-06-03-exam#startsFrom
:startsFrom rdf:type owl:ObjectProperty ;
            rdfs:domain :Stage ;
            rdfs:range :City .


###  http://www.semanticweb.org/2020-06-03-exam#takesPartIn
```

```
:takesPartIn rdf:type owl:ObjectProperty ;
             rdfs:domain :Rider ;
             rdfs:range :Race .


### http://www.semanticweb.org/2020-06-03-exam#takesPlaceIn
:takesPlaceIn rdf:type owl:ObjectProperty ;
              rdfs:domain :IntermediateSprint ;
              rdfs:range :City .


#################################################################
#    Data properties
#################################################################

### http://www.semanticweb.org/2020-06-03-exam#cityName
:cityName rdf:type owl:DatatypeProperty ;
          rdfs:domain :City ;
          rdfs:range xsd:string .


### http://www.semanticweb.org/2020-06-03-exam#raceName
:raceName rdf:type owl:DatatypeProperty ;
          rdfs:domain :Race ;
          rdfs:range xsd:string .


### http://www.semanticweb.org/2020-06-03-exam#raceYear
:raceYear rdf:type owl:DatatypeProperty ;
          rdfs:domain :Race ;
          rdfs:range xsd:integer .


### http://www.semanticweb.org/2020-06-03-exam#riderName
:riderName rdf:type owl:DatatypeProperty ;
           rdfs:domain :Rider ;
           rdfs:range xsd:string .


### http://www.semanticweb.org/2020-06-03-exam#stageDistance
:stageDistance rdf:type owl:DatatypeProperty ;
               rdfs:domain :Stage ;
               rdfs:range xsd:integer .


### http://www.semanticweb.org/2020-06-03-exam#stageNumber
:stageNumber rdf:type owl:DatatypeProperty ;
             rdfs:domain :Stage ;
             rdfs:range xsd:integer .


#################################################################
#    Classes
#################################################################

### http://www.semanticweb.org/2020-06-03-exam#City
:City rdf:type owl:Class .
```

```
### http://www.semanticweb.org/2020-06-03-exam#IntermediateSprint
:IntermediateSprint rdf:type owl:Class .


### http://www.semanticweb.org/2020-06-03-exam#Race
:Race rdf:type owl:Class .


### http://www.semanticweb.org/2020-06-03-exam#Rider
:Rider rdf:type owl:Class .


### http://www.semanticweb.org/2020-06-03-exam#Stage
:Stage rdf:type owl:Class .


### Generated by the OWL API (version 4.2.8.20170104-2310) https://github.com/owlcs/owlapi
```