(CSE 344 2016 AU Final - Hard MapReduce Question)

c) Given a list of pixel values, encoded as `(R,G,B)`, compute a histogram for each of the three color channels, where each pixel value can range from 0 to 255. Return the histogram as key-value pairs, with the key being a string of either `"R"` (red), `"G"`, (green), and `"B"` (blue), and value being an array of 255 integers, where each entry indicates the number of pixels in the input pixels that has that value of R/G/B. For instance, given this input of three pixels:

```
(3, 2, 3), (4, 2, 5), (3, 1, 5) // each pixel is of the form (r, g, b)
```

You should output:

```
("R", [0, 0, 0, 2, 1, 0 … 0]) // 255 entries total in array
("G", [0, 1, 2, 0 … 0])
("B", [0, 0, 0, 1, 0, 2, 0 … 0])
```

Note that `R[0] = 0` since none of the input pixels has red value equals to 0, but `R[3] = 2` since two of the input pixels has red value equals to 3.

(Problem 5, continued)

Write a **single stage** map reduce program in Hadoop that computes the output above. You can assume that the input is already parsed and is given to you as a list of `Pixel` objects. You can assume that the Hadoop `emit` method handles object serialization. Use only the Hadoop constructs that we discussed in class.

Do not worry too much about getting Java syntax correct, but make sure you clearly state what key-value pairs are emitted by `map` and how they are reduced. (17 points)

```
void map (Pixel [] pixels) {
// p.r accesses the red value of pixel p as an integer, and similarly for
// p.g and p.b









}

// fill in the method signature of reduce

void reduce (                                        ) {









}
```

(Problem 5, continued)

d) Explain, in a few words, why your solution above performs better than the naive implementation of having the mappers broadcast the entire `pixels` array to all reducers (Obviously you will get no points for c) if that was what you wrote).
(5 points)

(CSE 344 2017 AU Midterm - Hard SQL Question)

A company maintains a database about their employees and projects with the following schema.

Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)

WorksOn records which employee worked on which projects. salary and budget represent yearly salary and budget respectively. An employee may work on multiple project during the same year, and may also work on the same project during multiple years. All keys are underlined, and WorksOn.eid, WorksOn.pid are foreign keys to Employee and Project respectively.

(b) (10 points) We say that an employee worked intermittently on a project p if she worked on p during one year, then did not work on p during a later year, then worked again on p during an even later year. For example if Alice worked on the project during 2012, 2013, and 2017 then we say that she worked intermittently on p; if Bob worked on that project during 2013, 2014, 2015 and no other years, then we say he worked continuously. Write a SQL query to retrieve all employees that worked intermittently on some project. Return the employee name, and the project title.

(CSE 344 2017 AU Midterm - Hard Datalog Question)

Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)

(b) (10 points) A project p1 influences a project p2, if there exists an employee who worked on p1 during some year, then worked on p2 during some later year. After a major bug was discovered in several projects, the company traced it down to a design flaw in the "Compiler" project, and now wants to retain only the projects that were not influenced by "Compiler". (Note "Compiler" is influenced by "Compiler".) Write a datalog program to find all projects who were not influenced by the "Compiler" project; return their pid and title

(CSE 344 2016 AU Final - Hard Transaction Question)

```
T1: R(A), W(B), I(D), R(C)
T2: R(B), R(D), W(C)
T3: R(D), R(C), R(D), W(A)
```

d) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **both shared and exclusive table locks**? If so write such a schedule with lock / unlock ops, and explain why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking. (5 points)

e) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **only exclusive table locks**? If so write such a schedule with lock and unlock operations and indicate why the transactions are deadlocked. Otherwise write "No". Use `L1(A)` to refer to `T1` locking table `A`, and `U1(A)` for unlocking. (5 points)

(Problem 3 continued)
Transactions copied here for your reference.

```
T1: R(A), W(B), I(D), R(C)
T2: R(B), R(D), W(C)
T3: R(D), R(C), R(D), W(A)
```

f) Does there exist a schedule of the above transactions that would result in a manifestation of the phantom problem under non-strict 2PL with exclusive table locks? If so write out such a schedule with lock and unlock operations and indicate why there is a phantom problem. Otherwise write “No”. (5 points)

g) Suppose we change locking granularity to tuple rather than table level, where we only lock the tuples that are read / written / inserted from the affected table. Is there a schedule of the above transactions that would result in a manifestation of the phantom problem under non-strict 2PL with exclusive tuple locks? If so write out such a schedule with lock and unlock operations and indicate why there is a phantom problem. Otherwise write “No”. (5 points)

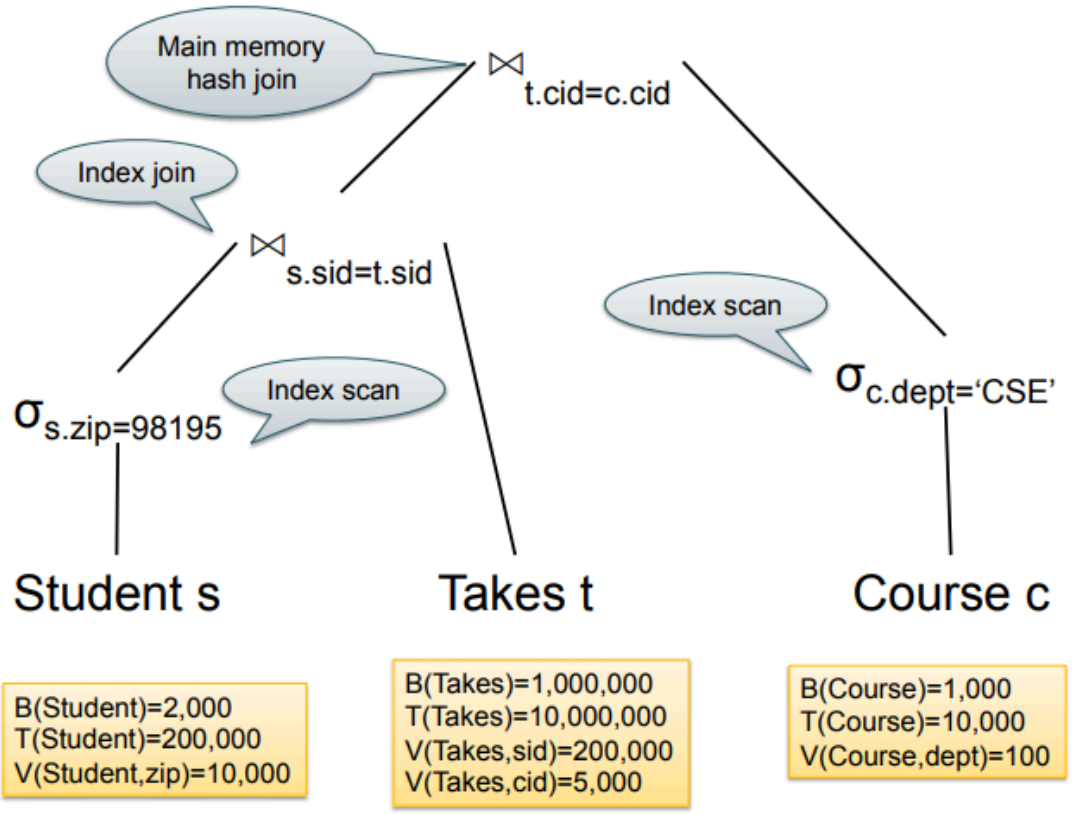
(CSE 344 2017 AU Final - Hard Cost Estimation Question)

Consider the relations below:

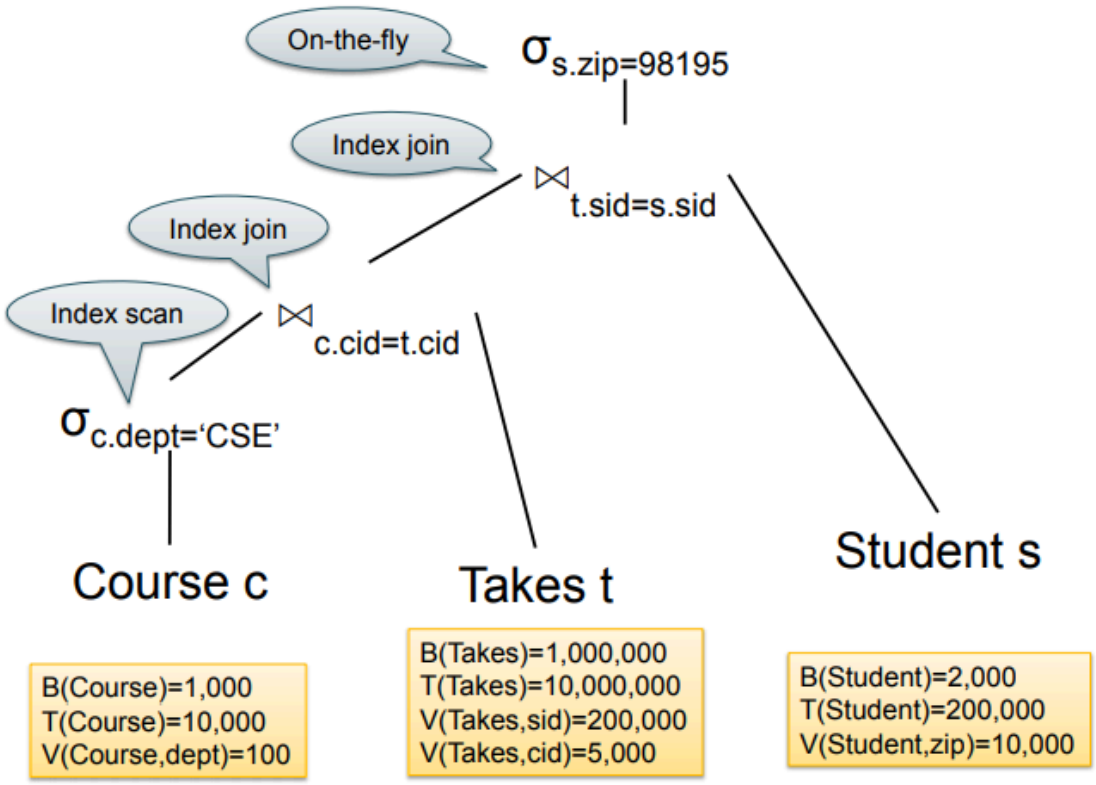
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)

(c) For each of the physical plans below, estimate its cost. Assume the size of the main memory is M = 2000 pages and that all indexes are unclustered, and are stored in main memory (hence accessing them requires zero disk I/O's).

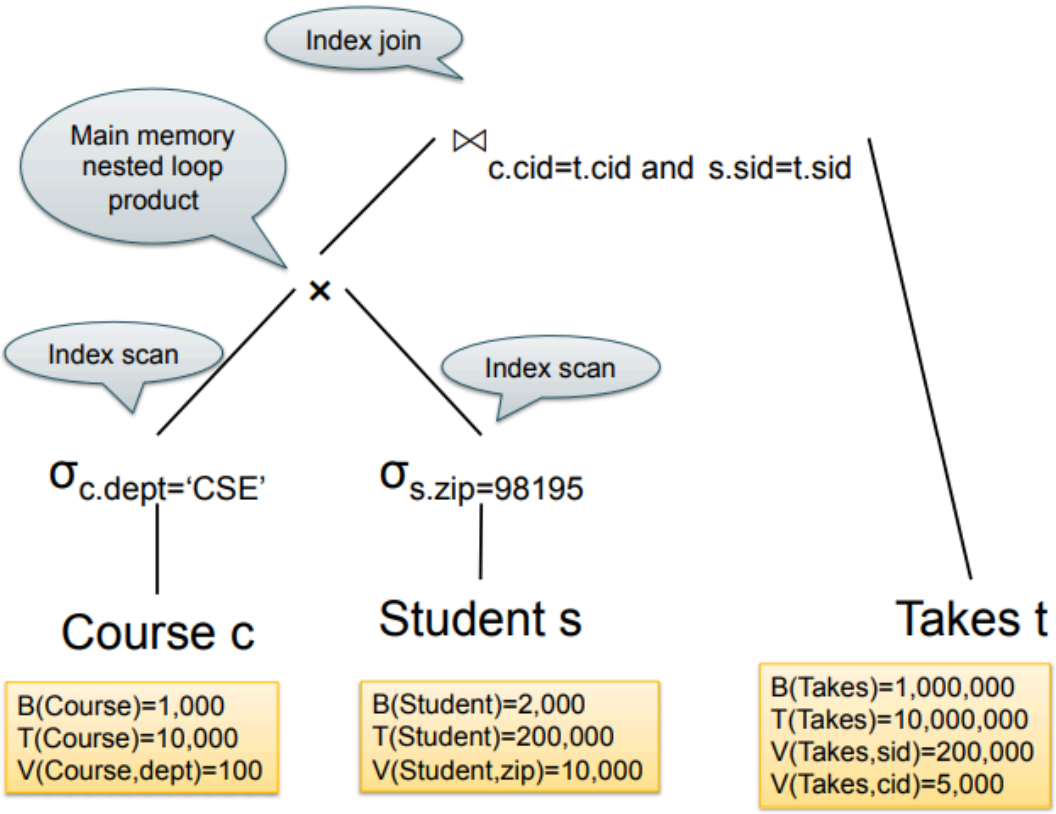
i. (5 points)



ii. (5 points)


On-the-fly
σs.zip=98195
Index join
⋈t.sid=s.sid
Index join
Index scan
⋈c.cid=t.cid
σc.dept='CSE'
Course c
Takes t
Student s
B(Course)=1,000
T(Course)=10,000
V(Course,dept)=100
B(Takes)=1,000,000
T(Takes)=10,000,000
V(Takes,sid)=200,000
V(Takes,cid)=5,000
B(Student)=2,000
T(Student)=200,000
V(Student,zip)=10,000

iii. (5 points)


Index join
Main memory nested loop product
⋈ c.cid=t.cid and s.sid=t.sid
×
Index scan
Index scan
σc.dept='CSE'
σs.zip=98195
Course c
Student s
Takes t
B(Course)=1,000
T(Course)=10,000
V(Course,dept)=100
B(Student)=2,000
T(Student)=200,000
V(Student,zip)=10,000
B(Takes)=1,000,000
T(Takes)=10,000,000
V(Takes,sid)=200,000
V(Takes,cid)=5,000