[Optional] Mini-Project

Start Assignment

Due Monday by 11:59pm **Points** 50 **Submitting** a file upload **File Types** zip
Attempts 0 **Allowed Attempts** 1
Available Jul 12 at 11:59pm - Aug 13 at 11:59pm about 1 month

In this optional Mini-Project, performed in groups of size two, you will practice applying your **Query Optimization** knowledge. This will include knowledge of:

- Cost-based Optimization
- Dynamic Programming
- RA Equivalences
- Heuristics

Estimated required time: 12-24 hours*

*The estimated time might be different based on team work assignments and familiarity with C++.

Team: This project is aimed to be performed in teams of two students. This is because of the estimated required time for the project (we do not want to overwhelm you in the last four weeks of the semester), as well as for the opportunity for teamwork and acquiring soft-skills. The project is optional and you do not have to do it. There are other minor individual programming elements incorporated in the homework assignment and interactive sessions if you do not wish to perform a group mini-project. However, if you want to do the project, it needs to be as a part of a team. There will be no grades associated with team formation, but only submissions from a group of two students are accepted. The groups are open to self registration under Mini-Project **team** form.

Description:

You need to implement your query optimization as a simplified version of join order selection, with the following conditions:

- Only natural join operation is required (inner, equality, common attribute name)
- Consideration of indexed access paths are not required
- Considerations of key attributes and the associated effects is required
- Consider memory buffer available as required
- When information not available, you can make definitions and assumptions about
  - Reduction factors
  - Range of attribute values
  - Size of attributes
  - Duplicate tuples

  However, these should not be random and there should be some reasoning about it throughout your

code.

- You not not need to consider the cost of writing the final result out

Note: This could be basically considered an implementation of an extension to the pseudo-code provided in Question (5) of Homework (3).

Input: file ***input.txt*** with the following format:

In the first line, there is one number n: Number of relations Ri in the join

Then the file follows with n lines, each line including the following, space separated:

Pi: Number of Pages in the relation Ri
Mi: Number of Tuples in the relation Ri
Ni: Number of Attributes in the relation Ri
List of attributes in the relation Ri, each attribute followed by the number V(Ri, attribute), the value count for that attribute
Ki: Number of Key Attributes in the relation Ri
List of attributes in the primary key of relation Ri

For example, please find sample ***input.txt***. This input file has total of three relations. The first relation considered R1, has 100 pages, 2000 tuples, and has 5 attributes that are named A, B, C, D, and E. The number of unique values for attribute A is 2000, for attribute B is 30, for attribute C is 10, for attribute D is 10 and and for attribute E is 100. The relation has a primary key that consists of attribute A.

The second relation named R2 has 50 pages, 1000 tuples, and 4 attributes. The attributes are A, with 1000 different values, S with 10 different values, and N with 20 different values, and M with 10 different values. This relation has one primary key attribute which is A.

The third relation named R3 has 100 pages, 300 tuples, and 3 attributes. The attributes are A, with 300 different values, R with 30 different values, and H with 40 different values. This relation has primary key attribute A.

Output: file ***output.txt*** with the following format:

the selected order of joins as a form of levels of a join tree (graphically it is reversed), followed by the cost of the operation in the line after each level, and the size estimation (carndinality) of the output at each level.

For example, please see ***output.txt*** for the provided input. As a caveat note that in the given set of input and output files, no information about foreign keys are given in the question. The size estimation is based on the assumption that the values for attribute A overlap. Remember from lecture videos where such information come from. If you make such assumptions, it should have a basis throughout your code, affecting your reduction factors. The cost of the second operation is based on the assumption that the result of the previous operation resides in memory and imposes no additional I/O. This could be inferred from given conditions in the project description.

Deliverables: Please submit a zip file including all your .cpp and header files.

Grading: The following will be considered in grading

- Algorithm for cost-based optimization
- Considerations for cost estimations
- Considerations for size estimations
- Heuristic rules applied
- Logical equivalence rules included
- Cleanliness and readability of code

Effect on Total Grade: As discussed in the first session of the semester and indicated in **grading**, the final grade is 35% of the total grade. This optional group Mini-Project can optionally be considered for 10% of this percentage. This will result in the weight of the final exam to be 25% and the Mini-Project to be 10% of your total grade. ***When you submit your optional Mini-Project, you commit to the 25% final exam and 10% Mini-Project grade structure regardless of the grades you receive in Mini-Project and Final Exam.***

Academic Honesty: Group work required. Parts of external code (with proper copy right and re-use permissions) only accepted if quoted and resource properly indicated. Similarity check will be performed on the submissions.