This Project is based on the SimpleDB database system. SimpleDB is a multi-user transactional database server written in Java. SimpleDB was written by Professor Dr. Edward Sciore and is widely used as a teaching tool for database systems. For this project, you will be implementing a new “join” algorithm and compare the cost of that with the existing “cross join” algorithm that comes with the SimpleDB.

The current join operation (referred to as the “product” as in relational algebra) that is implemented in the SimpleDB is a simple “cross join” or nested loop join. Your task is to implement another join algorithm. It can be any of the following three join algorithms-

- Block Nested Loop Join
- Merge Join
- Hash Join

Once you have implemented a new join algorithm, estimate the cost for the same queries and same tables using your join and the original nested loop join. Compare the cost and, and see how this compares with run times and I/Os incurred when you actually run the query.

N.B. For Merge-Join, you need to implement your own sorting algorithm, although you can make use of libraries for sorting within a block.

Project Description:

The tasks for this projects are-

Task 1 (Implement): Pick one of the three join algorithms mentioned above and implement it for the SimpleDB. Your implementation should work for any schema that the original implementation works for. Also, add a description of your implementation technique in your report.

Task 2 (Hand-done cost estimates): Hand-do the cost estimation for a query involving at least three tables and two joins. Do this for both your implemented algorithm and the simple nested loop already in SimpleDB, for the two possible join orders. This cost analysis with all intermediate steps should be included in your report.
Task 3 (Evaluate performance): Now, compute the costs using your code. There is a function, (blocksAccessed()) in simplesdb.plan.Plan.java the can help you to calculate the cost. After you have both versions of the cost estimates (hand done and machine computed), analyze how do the machine computed costs seem to compare with your hand done estimates. Add histograms or another form of graphs to represent your result in the report. Also, add an analysis of your results.

Task 4 (Compare): For this last task, we ask you to collaborate with at least one other member from your PSO who has implemented a different join algorithm. Come up with a scenario where yours should "win", do cost estimates, then run and see how those work. This will be similar to tasks 2 and 3, except comparing with your team member rather than the provided simple nested loop join. Each team can have 2-4 members. If you decide to have 4 members, then two of the team members can implement the same join algorithm. This may be, but need not be, the same team as last time (we’d encourage you to meet someone new.)

Project Deliverables:

1. Your Code along with test cases, and a ReadMe file using turnin (just like Project 1, except this time it is turnin -c cs448 -p project2 <submission folder>. Note that you need to connect to one of the CS lab machines, there are several ways of doing this. If you are on campus, you can use VPN to use resources as if you are on campus. https://www.itap.purdue.edu/connections/vpn/
2. A project report that covers all the points that are mentioned in the project description submitted in gradescope.

For each student, Task 1 to Task 3 must be done individually, although you can discuss your ideas with others. You will need to coordinate with your team members to ensure that you pick different join algorithms (even if you both feel the same approach will give the biggest improvement). Only task 4 will be the involve multiple team members. The report should be written only by yourself!!

Additional Instructions:

- Submit your team (2 to 4 members per team) to your PSO instructor by next Tuesday,
- If you cannot find a teammate, submit that to your PSO instructor who will assign you teammate(s). Note that you may end up having someone added to your team, even if you already have two or three.
- You can pick your own teammate but they need to be from the same PSO.
Some important Notes:

1. While doing task 3, remember to refresh your cache before computing the cost every time.
2. Implement the join algorithm in the original code base given to you for project 1. Use the default buffer manager for our testing purpose.
3. We will provide you with some modified parts of the code and some pointers to where you should be writing your modifications (see below). You can access that in BrightSpace.
4. The description may seem simple, but you may want to start early as this project requires significant coding on your part!!

Code Reference

For this project, your main implementations will be inside simpledb.query and simpledb.plan. You will write your new join algorithm inside simpledb.query. Once you have implemented that, you need to add a plan for your new join algorithm inside simpledb.plan.

We have provided you with some supporting code (in Brightspace, content / Project 2). Do the following steps before getting started with your implementation.

1. Add the given ProductScan2.java in simpledb.query. ProductScan.java has implemented the cross join in it, you will implement your own join algorithm in ProductScan2.java.
2. Add the given ProductPlan2.java in simpledb.plan. The current ProductPlan.java has plans implemented for the ProductScan.java. You will implement necessary computations, cost calculations for your implemented ProductScan2.java, inside ProductPlan2.java.
3. Replace the current BasicQueryPlanner.java in simpledb.plan with our given one
4. Replace the current OptimizedProductPlan.java in simpledb.plan with our given one
5. Fully understand ProductScan.java, ProductPlan.java, OptimizedProductPlan.java and BasicQueryPlanner.java before you start implementing.
6. Add PlannerTest3.java in simpledb.plan
7. Use PlannerTest2.java and PlannerTest3.java for initial testing. Make sure that everything builds and runs before you start working.
8. PlannerTest3.java has three tables and only one join. You will need to modify this test file as per your evaluation requirement stated in the project description (for the two join orders).