CS-245 Database System Principles – Winter 2002
Assignment 5

Due at the beginning of class on Tuesday, February 26

- State all assumptions and show all work.
- Subscribe to cs245@lists.stanford.edu to receive clarifications and changes.
- You can email questions to cs245-staff@cs.stanford.edu

Problems 1-3.

For the first three problems, use the following schema, for an on-line bookstore (this is the same schema from assignment 4):

```
Cust (CustID, Name, Address, State, Zip)
Book (BookID, Title, Author, Price, Category)
Order (OrderID, CustID, BookID, ShipDate)
Inventory (BookID, Quantity, WarehouseID, ShelfLocation)
Warehouse (WarehouseID, State)
```

This schema represents customers (Cust) and books (Book). When a customer buys a book, a tuple is entered into the Order table. The inventory of books is recorded in the Inventory table, which records the quantity, warehouse and shelf location for each different book. The Warehouse table records the state where each warehouse is located in. You may assume that these relations are sets. You may assume Price is a numeric value. You may also assume that ShipDate is an integer representation of a date. Finally, you may assume that there is a constant, `:today`, that contains the integer representation of today's date.

Assume you have an index over `Book.Author`, but there are no other indexes.

Problem 1. (30 points)

For each of the following logical query plans (written as relational algebra expressions) construct two physical query plans that produce the same query result. One of your plans should be likely to be more efficient than the other. State briefly (one or two sentences) why one plan is likely to be more efficient. (It is not necessary to count I/Os or tuples, but only to give a qualitative assessment why the plan is more efficient.) Assume that intermediate results produced by one operator are written to disk to be read back into memory by the next operator that needs them. You may use the following physical operators:

- Table-Scan
- Index-Scan
- One-Pass-Selection
- One-Pass-Projection
- One-Pass-Join
- Two-Pass-Hash-Join

a. $\sigma_{Author='Mark\ Twain'}(Book)$

b. $\pi_{Title,Author}(Book)$

c. $\pi_{Title,Author,BookID}\{ [(\sigma_{ShipDate=:today} (Order)) \bowtie (\sigma_{Category='Mystery'} (Book))] \bowtie [\sigma_{State='CA'} (Cust)] \}$

Problem 2. (40 points)

Indicate the number of I/Os required to perform the following operations. You should assume that each operation uses memory efficiently. You can ignore final output I/O cost. If required, explain briefly. For this problem, use the following statistics:

- B(Order) = 3,000 blocks
- B(Cust) = 1,000 blocks
- B(Book) = 100 blocks

a. Selection of Price<10 over Book. Memory= 10 blocks.

b. One pass join of Order and Cust. Memory=1001 blocks.

c. Nested loop join of Order and Cust. Cust is the outer relation. Memory=2 blocks.

d. Nested loop join of Order and Cust. Cust is the outer relation. Memory=101 blocks.

e. Nested loop join of Order and Cust. Order is the outer relation. Memory=101 blocks.

f. Nested loop join of Order and Book. Book is the outer relation. Memory=11 blocks.

g. Hash join of Order and Book. Memory = 11 blocks. (You may assume that you have a hash function over Order.BookID and Book.BookID that distributes the tuples evenly into equally sized buckets.)

Problem 3. (25 points)

For the join Order $\bowtie$ Cust, state the minimum number of memory blocks you would need to perform each of the following join algorithms. Again, you may assume B(Order) = 3,000 blocks and B(Cust) = 1,000 blocks.

a. One pass join

b. Nested loop join

c. Two pass simple sort join

d. Two pass Sort join (also known as sort-merge join)

e. Two pass hash join

Problem 4. (5 points)

Query processors tend to use heuristic query optimization. That is, the optimal query plan is not found, but heuristics are used to generate a plan that is not “too bad.” In 100 words or less, sketch an algorithm for finding the optimal physical query plan for a given logical query plan. Are DBMS vendors likely to adopt your algorithm? Why or why not?