Solutions Problem Set 7

Part 1

15.3.2

a) The answer is 120,000. The reasoning is as follows:
M-1= 999 so we can only load 999 blocks of one relation (R) in the outer iteration. We have to first do this 10 times, and for each of those times the 10,000 blocks of S get loaded. Hence the cost so far is 10*(10,000+999)=109,990.
The last iteration is special because only 10,000-9990 blocks of R or 10 blocks remain. Hence, in the final iteration we load the last 10 blocks of R and again the 10,000 blocks of S so we have 10,010 IO. Adding this to the above number, we get 120,000.
Many people blindly applied the formula in the textbook. That formula usually only holds if M-1 divides one of the B's exactly. In general, always reason your way through.

15.3.3

a) This was a tough problem. The answer is M=1250. To see why, consider that M-1=1249. Hence, we would have to read R in 8 times (in blocks of 1249), and for each of those times, we would load S in once, giving disk IO (1249+10,000)*8=89992. Next, we load in the remaining blocks of R which is 10000-9992=8. We also need to load in the 10,000 blocks of S again for this last 9th iteration. This gives us 10,000+8 disk IO. Adding the two, we get disk IO of exactly 100,000.
If you follow the same procedure for a smaller M, you'll find that you'll need more than 100,000 disk IO. For larger M you would need more.
You might ask how we came up with the answer 1250. It's the solution of a non-linear equation that I'll be happy to derive in the review session or office hours. The clue is in following the same procedure above but with M as a variable. M may not exactly divide one of the B's, which will give you a non-linear equation.

c) It's impossible to do a nested loop join of two relations of size 10,000 in only 15000 Disk IO's! Even if we had a memory of size 20,000, we would still have to load both relations in memory at least once, which has a minimal cost of 20,000. Again, if you tried applying some formula, it would have given you some positive answer, but you can clearly see it would be wrong.

15.4.2

a) It is simply 3(B[R]+B[S])=60,000

b) 5(B[r]+B[s])=100,000

16.2.2

a) Consider R(a,b) and an instance {(3,2),(2,3)} and {(2,4),(2,3)}. If we project b first from both and do set union we will get {2,3}. However, if we first do set union and then project b we get {2,2,3} and the two are not the same.
The definition of a set union might be a little confusing here. A good way to think about it is first to do a bag union and then apply duplicate elimination. It is not correct to assume that the result is a permanent set (so that duplicates keep getting eliminated no matter what the operator further on) but that right after the set union, the result (formally a bag) is also a set.

c) Consider {(2,3),(2,4)}. Projecting b first and then eliminating duplicates will simply give {2}. Eliminating duplicates first (there are none) and projecting b would give {2,2}. Note that here we are dealing with bags so the two are not equal. For sets, the two ARE equivalent!

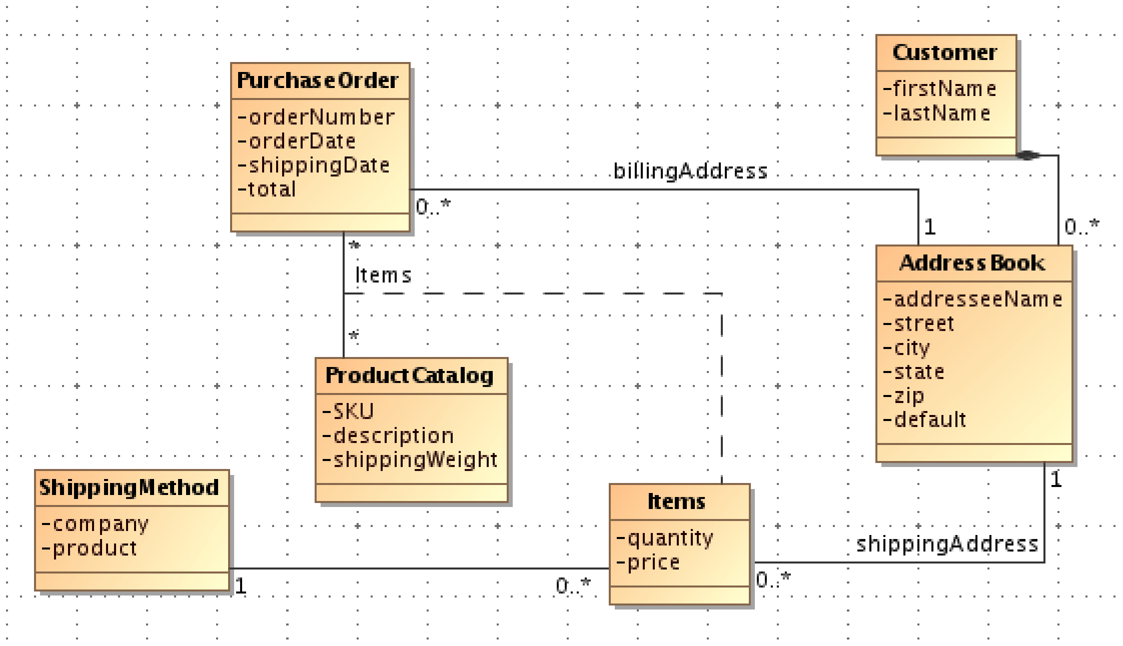
16.2.6
a)

$$\Pi_{b+c\rightarrow x,c+d\rightarrow y}(\Pi_{b,c}(R) \bowtie \Pi_{b,c,d}(S))$$

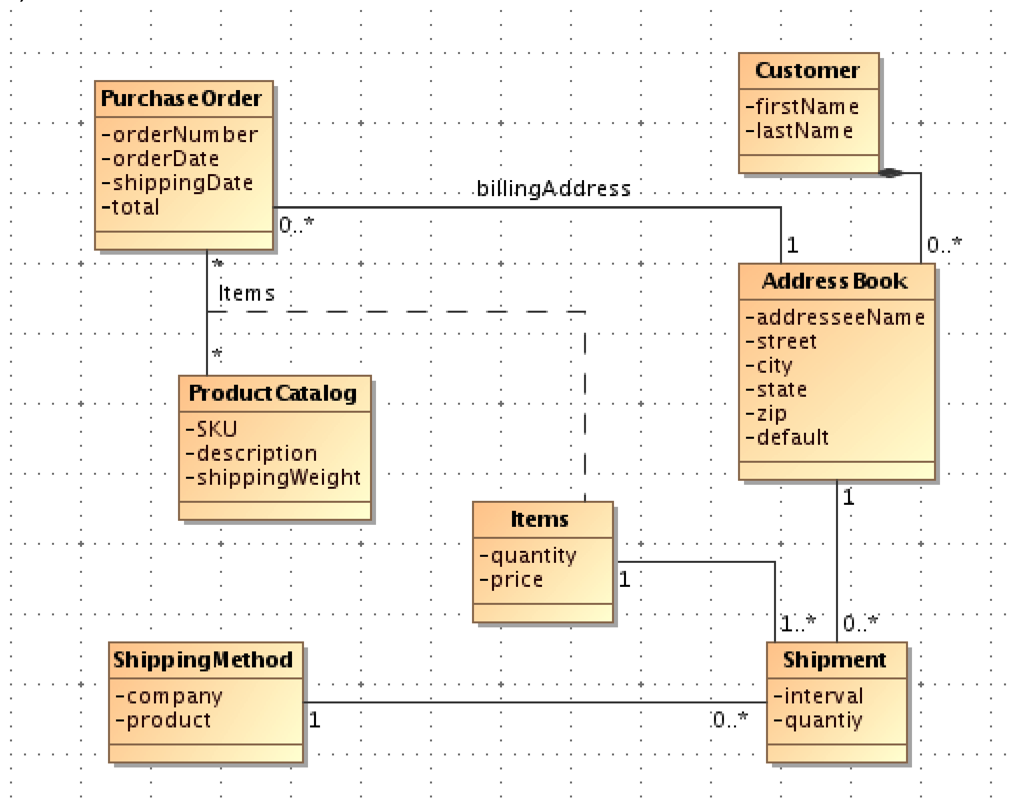
b)

$$\Pi_{a,b,a+d\rightarrow z}(R \bowtie \Pi_{b,c,d}(S))$$

Part 2
1)



2)


PurchaseOrder
-orderNumber
-orderDate
-shippingDate
-total
Customer
-firstName
-lastName
billingAddress
0..*
1
0..*
Address Book
-addresseeName
-street
-city
-state
-zip
-default
*
Items
*
Product Catalog
-SKU
-description
-shippingWeight
Items
-quantity
-price
1
1
1..*
0..*
ShippingMethod
-company
-product
1
0..*
Shipment
-interval
-quantiy