NAME: Xiang Xu

605.741.31
Module 6 Quiz

1) Assume the following tables and attributes:

CUST:	cid	- int	(4 bytes)	
	name	- char(20)	(20 bytes)	
	age	- int	(4 bytes)	- range 21 to 60
	address	- char(50)	(50 bytes)	
	state	- char(2)	(2 bytes)	
ITEM:	cid	- int	(4 bytes)	
	itemid	- int	(4 bytes)	
	price	- int	(4 bytes)	- range $1 to $100
	description	- char(50)	(50 bytes)	

- CUST is located on site 1; ITEM is located on site 2.
- There is a one-to-many relationship between CUST and ITEM where cid is the foreign key of CUST (on cid) in the ITEM table.
- CUST has hashed indexes on cid.
- There is NO index on the ITEM table.
- There are 100 CUST records and 1000 ITEM records.
- T_{TR} is the network transfer cost per byte.
- T_{CPU} is the cost of a CPU instruction (including disk I/O)
- T_{MSG} is the cost of initiating and receiving a message.
- There are only two distinct states in the CUST table records (MD and VA).

Given the query:

```
SELECT  c.name, c.state, i.description
FROM    CUST, ITEM
WHERE   CUST.cid = ITEM.cid and
        CUST.age > 50 and
        ITEM.price > 30 and
        ITEM.price <= 50
```

See next pages for questions.

a) Estimate the total cost of the query by moving the appropriate parts of the CUST table from site 1 to 2 and performing the join at site 2. To estimate the total cost, sum the cost of the estimates of each intermediate step. Assume that you are done when you have completed the join. (This is NOT a semi-join.)

Cost =
T_CPU* card(Customer) = T_CPU * 100 rows

//read Customer 100 rows
//keep only Customers of Age >50. Estimate to be (60-50)/(60-20) = 1/4
//transfer a table Customer' which contains columns cid, name and state where Age > 50
//row of Customer' is 100 * 1/4 = 25 rows

+ T_MSG + T_TR * size(Cutomer') = T_MSG + T_TR * 650 bytes

//transmitting the rows of Customer' to site 2
//cid + name + state = 4 + 20 + 2 = 26 bytes
//Customer' 25 rows * 26 bytes = 650 bytes

+ T_CPU* card(Customer') = T_CPU * 25 rows

//writing Cutomer' to the site 2

+ T_CPU* card(Customer') + T_CPU* card(Customer')* card(ITEM) +
T_CPU* card(ITEM) * SelectivityFactor_Age > 50
= T_CPU* 25+ T_CPU* 25 * 1000 + T_CPU* 1000 * 1/4
= T_CPU (25 + 25000 + 250)
= T_CPU * 25275

//Cost of join (full table scan)
//*Assume that you are done when you have completed the join.*

Cost =

T_CPU* card(Customer) = T_CPU * 100 rows
+ T_MSG + T_TR * 650 bytes
+ T_CPU * 25 rows
+ T_CPU * 25275

=T_CPU (100 + 25 + 25275) + T_MSG + T_TR * 650
=T_CPU * 25400 + T_MSG + T_TR * 650

b) Estimate the total cost of the query by moving the appropriate parts of the ITEM table from site 2 to 1 and performing the join at site 1. To estimate the total cost, sum the cost of the estimates of each intermediate step. Assume that you are done when you have completed the join. (This is NOT a semi-join.)

Cost =
T_CPU* card(ITEM) = T_CPU * 1000 rows

//read Item 1000 rows
//keep only Item of price > 30 and price <= 50. Estimate to be (50-30)/(100) = 1/5
//transfer a table Item' which contains columns cid, description where price > 30 and price <= 50
//row of Item' is 1000 * 1/5 = 200 rows

+ T_MSG + T_TR * size(Item') = T_MSG + T_TR * 10800 bytes

//transmitting the rows of Item' to site 1
//cid + description = 4 + 50 = 54 bytes
//Item' 200 rows * 54 bytes = 10800 bytes

+ T_CPU* card(Item') = T_CPU * 200 rows

//writing Item' to the site 1

+ T_CPU* card(Item') + T_CPU* card(Item') +
T_CPU* card(Customer) * SelectivityFactor_30<Price<=50
= T_CPU* 200+ T_CPU* 200 + T_CPU* 100 * 1/5
= T_CPU (200 + 200 + 20)
= T_CPU * 420

//Cost of join (hashed index)
//*Assume that you are done when you have completed the join.*

Cost =
T_CPU * 1000 rows
+ T_MSG + T_TR * 10800 bytes
+ T_CPU * 200 rows
+ T_CPU * 420

= T_CPU * (1000+200+420) + T_MSG + T_TR * 10800
= T_CPU * 1620 + T_MSG + T_TR * 10800

c) What single modification to the database would enable an even lower cost query plan? (I am not looking for a semi-join.)

Solution a) is very slow, because it is doing a cartesian product on the join
Solution b) is transferring too many bytes over the network

the single modification is to add a hashed index on the cid column in relation Customer, this change will improve the performance a lot, I have the updated cost of the solution a) after we applying the hashed index.

+ T_CPU* card(Customer') + T_CPU* card(Customer')~~* card(ITEM)~~ +
T_CPU* card(ITEM) * SelectivityFactor_Age > 50
= T_CPU* 25+ T_CPU* 25 ~~* 1000~~ + T_CPU* 1000 * 1/4
= T_CPU (25 + 250~~00~~ + 250)
= T_CPU * ~~25275~~ 300

//Cost of join (hashed index)
//*Assume that you are done when you have completed the join.*

Cost =
T_CPU* card(Customer) = T_CPU * 100 rows
+ T_MSG + T_TR * 650 bytes
+ T_CPU * 25 rows
+ T_CPU * 300

=T_CPU (100 + 25 + 300) + T_MSG + T_TR * 650
=T_CPU * 425 + T_MSG + T_TR * 650