**Final Report Mini-Project 3**

**CS522**

**by Aaron Barr, Suyog Mankar, Kevin Leehan**

**Introduction**:

The centralized database we used was that of a third party managing products from different companies and keeping track of customer reviews of those products. The scenario we have come up with involves multiple branches of a third-party store where each branch offers the same products as other branches, but may have different customer bases. The corporate headquarters is keeping track to analyze the information.

**Server Site Analysis:**

The number of sites required for this type of scenario are 3, one for the corporate offices and 2 for the local branches. The corporate offices will have the tables: Review, Product, Company, CompanyHasProduct, and ThirdParty. The local branches will have the tables: Product(replicated), Buy(fragmented), and Customers(fragmented).

**Schema:** The Global External levels would be individuals working at a corporate headquarters. The Conceptual level will allow the global users access to the tables that are not stored at the headquarters. The fragmentation schema has the Customer and the Buy tables each fragmented horizontally into 2 parts and each part is kept at a local branch. This allows for each store to keep track of their own customers and the number of sales, while allowing the corporate headquarters to access them if necessary. The corporate side of handling data from companies that supply them is kept at the corporate site.

**Queries and Query Analysis:**

**1)**Display all products and their average rating, product and review will be joined on product id.

Select Prod\_Name AS Item, AVG(R\_Rating) AS "Average Rating"

FROM Product Join Review

ON Product.Prod\_Id = Review.Prod\_Id

GROUP BY Prod\_name;

**A)** The tables for this query will be exclusively on the global users site. To optimize the execution of this query we would need a B-tree on Products.

**B)** The communication cost of this solution would be 0 bytes transferred because the query would run on the system of the global user.

**2)**Add a new product to the list of Products. (Requires a trigger to update the duplicates of the table)

INSERT INTO PRODUCT VALUES( '8305','Car','1000');

SELECT \* FROM Product

MINUS

SELECT \* FROM Product@LinkA;

SELECT \* FROM Product

MINUS

SELECT \* FROM Product@LinkK;

**A)** This query doesn’t need a B-tree because the Tables should be identical so using the MINUS operation will be faster than without it.

**B)** The communication cost of above query will be 1,433,810 byte transferred between the 2 queries.

**3**)Find the amount of sales of all Products for all time

Select Prod\_Name AS Item, Count(Prod\_Id) AS "Total Sold"

FROM Product Join

(SELECT Prod\_Id FROM Buy@LinkK

UNION ALL

SELECT Prod\_Id FROM Buy@Linka

ORDER BY Prod\_Id)

using (Prod\_Id)

group by Prod\_Name;

**A)** Each “Buy” table from the local branches needs to be sent to the global users site and then their Union can be handled by the global user. A B-tree for Products on Product\_Id would allow for a faster execution of the join.

**B)** The communication cost of this query will be 133,035 bytes transferred (5bytes \* 26,607 total records)

**4)** Display all combined customer information from site A and site B.

SELECT \* FROM Customer@LinkA  
 UNION  
 SELECT \* FROM Customer@LinkK

**A)**  We can’t use B-trees for this query because the information will be coming from other sites and compiled at the global user.

**B)**The communication cost of the query would be 3,563,541 bytes transferred.

**5)**  Display review information and associated product name joined from product table as well as customer associated name joined from associated site customer table.

SELECT Cust\_Name, R\_DATE, Prod\_NAME, R\_COMMENT, R\_RATING   
 From Product Join

(Review Join   
 (SELECT Cust\_Id, Cust\_Name FROM Customer@LinkA   
 UNION   
 SELECT Cust\_Id, Cust\_Name FROM Customer@LinkK   
 ORDER BY cust\_id)   
 USING (cust\_id))  
using (prod\_id)   
order by r\_date

**A)** A B-tree on the Review table will allow for faster matching of the Unioned Customer table and A B-tree on the Product Table will do the same for the second join.

**B)** The communication cost of query would be 1,147,581

**Record Totals for each Table Used in the Queries**

Product = 20,483

Buy(Frag 1) = 11,762

Buy(Frag 2) = 14,845

26,607

Review = 10,001

Customer(frag1) = 20,145

Customer(Frag2) = 40,254

60,399