**Homework 3**

Zhongyuan Fu

**Question1**

**A. Range-based partitioned on the age column**

Case 1:

Step 1: on each machine, do the following query:

Select city, Sum(age), Count(age) From Customer

Group By city;

Step 2: every machine sends its output to one machine

Step 3: on that machine,

for each city, compute the total sum of its age and the total count of its age

then we can get the Avg(age) by (total sum / total count)

Step 4: output city, Avg(age)

Case 2:

On each machine, just do the following query independently:

Select distinct age, city

From Customer

Case 3:

Step 1: Find the target machines which store the tuples with age above 30.

Step 2: Do the following query on each target machine independently:

Select \* From Customer

Where (city = ‘Boston’ or city = ‘New York’)

And age > 30

**B. Hash-based partitioned on the entire record**

Case 1: (the same with A.Case 1)

Step 1: on each machine, do the following query:

Select city, Sum(age), Count(age) From Customer

Group By city;

Step 2: every machine sends its output to one machine

Step 3: on that machine,

for each city, compute the total sum of its age and the total count of its age

then we can get the Avg(age) by (total sum / total count)

Step 4: output city, Avg(age)

Case 2:

Step 1: Re-partition the relation using a hash function on age and city

Step 2: So every machine creates m partitions and send the ith partition to machine i

Step 3: Do the following query on each machine:

Select \* distinct age, city From Customer

Case 3:

Because this is a hash-based (over the entire tuple) relation which is expected to be well distributed over all nodes, so we have to scan all the partitions.

Do the following query:

Select \* From Customer

Where (city = ‘Boston’ or ‘city’ = ‘New York’)

And age > 30

**C. Hash-based partition on the city column**

Case 1:

Just do the following query on each machine independently:

Select city, Avg(age) From Customer

Group By city;

Case 2:

On each machine, just do the following query independently:

Select distinct age, city

From Customer

Case 3:

Step 1: find the target machine which store the Boston and New York record

Step 2: on those machines, do the following query:

Select \* From Customer

Where (city = ‘Boston’ or city = ‘New York’)

And age > 30;

**Question 2**

1.

Step 1: select R.y by condition on R.x1

Step 2: send the output of step 1 to S’s location

Step 3: do join based on y column in S’ location (semi-join)

Step 4: send the records of S that will join (without duplicates) to R’s location

Step 5: perform the final join in R’s location

2.

Step 1: select the whole relation S by condition on S.z1

Step 2: send the output of step 1 to R’ location

Step 3: perform the final join in R’ location

3.

Scenario: either R.y or S.y has little duplicates and the the data in R.y and S.y are almost the same

**Question 3**

1. Purpose: Maintain the whole distributed database system in a consistent state. Reason: Each transaction has to be executed completely, which means it must accord with the database’ atomicity. Two - phase commit is one way to help database maintain its atomicity.
2. T will commit. The coordinator will send ‘commit T’ to every machine. After the crashed site is up again, it will communicate with the coordinator to know whether it is going to commit or rollback. There is a extremely situation that coordinator is crashed as well, then this site can go to communicate with other sites, to find out it will commit or rollback.
3. If the coordinator got the messages from all of the sites before it crashed, and all the messages are ‘ready T’, then it will send ‘commit T’ to all the sites after it is up again, but if at least one message is ‘don’t commit T’, then it will send ‘abort T’ after it is up again. However, if the coordinator do not get all the message from all of the sites before it crashed, then after it is up again, it will send ‘prepare T’ again to all the sites.