A Method for Secure Query Processing in Mobile Databases

D. Saha and N. Chowdhury

***Abstract*—A method for secure query processing in mobile databases has been presented in this paper. Mobile devices send queries to the server via point-to-point channels. The mobile database server computes the superset of results of several similar queries, which is broadcast on a large bandwidth broadcast channel. The mobile devices tune to that channel to retrieve their necessary information. Here the senders of all the said queries can view not only the result of their own query but also the results of others' queries in the same group. This is not desirable, since some user can maliciously view the results of queries sent by some other user. In this paper, we have proposed a method that would not allow any user to do so, although the results of multiple queries are being sent in the same broadcast channel which is accessible to all users.**

***Index Terms*— information security, mobile database, superset computation, query optimization..**

I. Introduction

Nowadays, there are various popular wireless devices such as cellular phones, personal digital assistants, and laptop computers with millions of users. The mobile computing environment has two parts, namely, mobile computers or mobile devices and a wired network of computers. In a mobile database the database and the corresponding database server lie in the wired network of computers referred to above, and the database clients lie in the mobile devices. Mobile devices send queries to the server via point-to-point channels. The mobile database server processes those queries and may send the results to the respective senders through the dedicated point-point channels.

But sometimes the results of the multiple queries have common portion and thus this portion of results are sent once for each query. This leads to inefficient utilization of the limited bandwidth of point-point channel. One way to solve this problem is to use a query optimizer/processor that performs query processing and optimization. Usually the mobile database server computes the superset of results of several similar queries, which is broadcast on a large bandwidth broadcast channel. In this work, we have proposed a method that would ensure that the result of the query sent by a user can not be viewed by the sender of other similar queries.

Mobile databases find application in electronic mail, public safety, stock trading, airline activities, weather information, bill paying, healthcare and the transportation industry [1]. A mobile database is a database that provides data to a mobile user [2,3], and the database may not be mobile itself. P. Sistla et al [4] propose a mobile database model where some data is present at the central server and some at the mobile devices, i.e. the data is distributed. In a distributed architecture there is a possibility of devices being disconnected or powered off, and thus not able to answer a request always. In one of our previous works on query optimization we have assumed a purely centralized architecture. In this architecture the whole geographical area is divided into cells. These cells are roughly circular in shape. At the center of each cell is a Base Station (BS), also referred to as a server, which communicates with the mobile devices in its cell area through the wireless channel. Each BS serves an area and those areas are connected via a wired network as referred to earlier. When a mobile device moves from one cell to another it begins communicating with the new BS in the new cell. In a centralized mobile database the database resides in the central server or BS. A mobile user can get data from the server in two methods: *pull-based* and *push-based*. In a *pull-based* method there are two channels namely an uplink channel and a downlink channel, which is also called the pull channel. A mobile device sends the query to the server via the uplink channel and the query results come to the device via the downlink channel. The downlink channels are private to each mobile device. In a *push-based* method the server broadcasts the data on a broadcast channel and the mobile devices tune to that channel to retrieve their necessary information [5]. In this case the server has to periodically broadcast information to the clients. In a *hybrid model,* the push-based method is extended by using an uplink channel via which clients can send explicit requests [6]. From the point of view of query processing we note that in a push-based method answers for each query are sent separately. But if there be common data among some of the query results it has to be sent once for each query. This results in poor bandwidth utilization. In this work we have attempted to use Multiple Query Processing to optimize bandwidth usage and have proposed a security measure in order to prevent malicious unauthorized access of the results of similar queries.

In mobile technology, wireless communication is used. In wired communication the transmission medium is a guided one and we use wires or optical fibers as the communication channel. Whenever we are in need of bandwidth we can lay

new set of wires or optical fibers to create a new physical channel that is totally isolated from all previous channels. But in the case of wireless communication the transmission medium is an unguided one. We cannot create new physical channels hence we cannot create bandwidth. When a new user registers he/she is allocated a portion of the existing bandwidth. Also in the mobile environment there is no practical upper limit on the number of users. So, the bandwidth is very scarce in the mobile environment.

We have surveyed several works on Multiple Query Processing and detection of common sub-expression [7,8,9]. These works do not deal with the mobile database environment and are applicable to the static environment only. Rajeshwari Malladi and Karen C. Davis [10] report that if there is subsumption between queries then they can be grouped for Multiple Query Processing and the result of the biggest query can be broadcast after sorting the same by a method called filtering of data [11] such that mobile devices can get their result by tuning to the broadcast channel at specific times. Thus this work takes care of only subsumption/equality. In the present work we have taken care of all types of commonality: subsumption/equality and overlap if it is of significant degree. The next section describes the proposed method in detail. Section 3 states the concluding remarks.

II. The Proposed Method

In the static environment the client machines are connected to the network via high bandwidth wires. So, in that case bandwidth is not that scarce. But in mobile databases we have to optimize the number of bytes transmitted for dispatching query results. So the query optimizer in this case has to take care of that requirement which is not the case with the static environment where optimization of the processing time is the main concern. The query optimizer incorporates Multiple Query Processing technique. But the results of multiple queries must be sent in the common broadcast channel in such a way that the sender of a specific query can get access to the result of his query only, not to the results of others' queries.

As stated above, in query optimization, we do not process each query separately but we process a group of queries together to take advantage of their similarity. Here Multiple Query Optimization is being used for computation of superset of results as discussed later.

To perform Multiple Query Processing we have to choose a group of queries and process them simultaneously. It may be noted that for the example database considered for experimentation in this work, each query requires a specific subset of tuples from each table in the group. Here we are identifying the set of tuples required by each query in a group from each table (in the group's table list) in that group. Then we are reducing each table (in the group's table list) to one that contains tuples that are required by at least one query in that group. Then the tables so computed are joined to compute the superset. While joining these tables we eliminate spurious tuples (not a part of any query's result in the group) that will arise in the process. After the completion of optimization we are left with an efficiently computed superset Σ of results (ρ_i , i = 1..N) for each query (q_i, i = 1..N) such that $\rho_i \varepsilon \Sigma$ for all i = 1..N and if a tuple $t \varepsilon \Sigma$ then there exists a query q_i such that $t \varepsilon \rho_i$. Then the superset is broadcast. An example database is shown below.

Table 1. The loan table (lno, bname, amt)

lno	*bname*	*Amt*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Bname	*bcity*	*assets*
Brighton	Brooklyn	7,100,000
Downtown	Brooklyn	9,000,000
Mianus	Horseneck	400,000
North Town	Rye	3,700,000
Perryridge	Horseneck	1,700,000
Pownal	Bennington	300,000
Redwood	Palo Alto	2,100,000
Round Hill	Horseneck	8,000,000

Table 2. The branch table (bname, bcity, assets)

A. The overview of the system and the proposed security measure

In this work we have adopted a hybrid model and assumed that there are two kinds of channels in the mobile environment, namely, a unidirectional (server-to-mobile devices) broadcast channel of large bandwidth and a number of point-to-point bi-directional channels of small bandwidth. The broadcast channel is used for dispatching query results and the point-to-point channels are used by mobile devices for sending queries to the server and by the server for sending device specific control information and encryption keys to the mobile devices. The point-to-point channels have been assumed to be secure private channels. This can be achieved by encryption or

technologies like CDMA. In this work we have focused on security of the shared channel. In this work we assumed a simple subset of SQL with the following format for the SQL query statements.

```
select <projection-list>
from <table-list>
where<predicate>
```

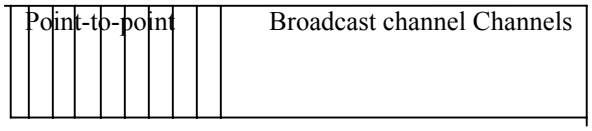

Fig. 1. Bandwidth allocation

The system performs its operation in the following steps.

1. Collection of queries.
2. Decomposition into groups.
3. Computation of supersets.
4. User specific encryption of the superset
5. Broadcasting the superset.
6. Extraction of the individual results.

1) Collection of queries

The optimizer is a multithreaded program. In this step the main thread of the program waits for a specific time window and collects all the queries arriving at the server. Then a new thread is created to perform query optimization while the main thread goes on collecting queries for another time window. This process goes on. The optimal span of the time window depends upon several system parameters: the size and capacity of the system, the load on the system, query arrival pattern etc.

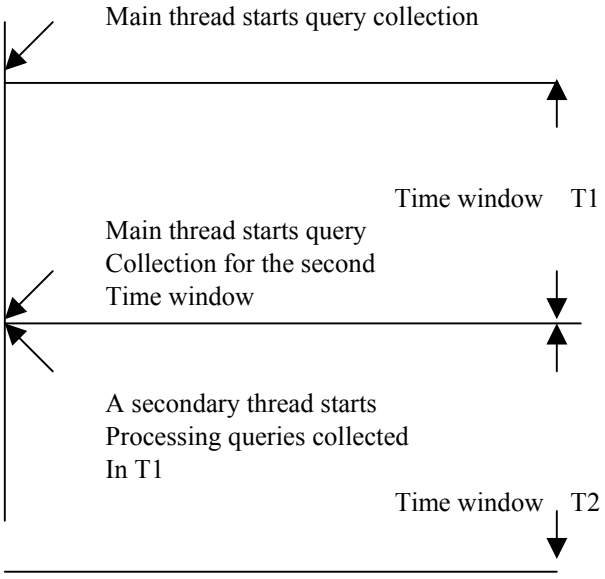

Fig. 2. The operation of the query optimizer

2) Decomposition into groups

Different queries have different structures. Some queries are best evaluated together while others are not. Based on structural similarity the queries collected in a specific time window are decomposed into groups such that the queries in a group have maximum possible commonality. The criteria used for group decomposition is the equality of the projection list and the table list so that the result of each query in a group has the same domain. For example, let us consider the following incoming queries in a specific time window.

Q1: select lno
from loan , branch
where amt between 900 and 1400 and assets >= 1700000 and loan.bname = branch.bname

Q2: select bname
from branch
where bcity = 'Brooklyn'

Q3: select lno
from loan , branch
where amt between 950 and 1500 and assets >= 2100000 and loan.bname = branch.bname

Q4: select bname
from branch
where bcity = 'Brooklyn'

The set of queries collected S = {Q1, Q2, Q3, Q4} is portioned into disjoined subsets G1, G2 and the like. The projection list for Q1 = {lno}, projection list for Q3 = {lno}. The table list for Q1 = {loan, branch}, table list for Q3 = {loan, branch}. From this we see that the queries Q1 and Q3 have the identical projection list and table list. Similarly the queries Q2 and Q4 have the same projection list and table list. So, there will be two groups G1 = {Q1, Q3} and G2 = {Q2, Q4}. The groups so formed are processed separately.

There will be queries that cannot be grouped with any other queries. For example, if only the queries Q1, Q2 and Q3 had arrived in this time window then Q2 could not be grouped with any other query. There can be multiple such queries. These queries are put in a separate group such that queries in that group are separately evaluated and results are separately dispatched. If in a group there is not much commonality then this procedure will lead to extra overhead and in that case we will get better results by processing and dispatching them separately. So we use a heuristic for measuring degree of commonality in a specific group.

The heuristic : For each attribute corresponding to a table we compute the superset of ranges of attribute values required by queries in a group. Query Q1 in G1 wants the range [900, 1400] for the attribute 'amt' whereas Q3 in the same group wants the range [950, 1500] for 'amt' from the table loan. So, the superset S of the attributes is [900, 1500]. We ensure that the range length of attributes for each table for each query must

be greater than 70% of the length of the superset for that table. This should hold for each attribute in the predicates of queries and for each table in the group otherwise we subdivide the group into smaller groups and if necessary some queries in the group are marked for separate processing and dispatching.

For Q1 in G1 and for attribute 'amt', the range length = 1400 – 900 = 500, the length of S = 1500 – 900 = 600. Therefore the percentage of commonality = 500/600*100 = 80%. For Q3 in G1 and for attribute 'amt', the range length = 1500 – 950 = 550 and percentage of commonality = 550/600*100 = 91.67%.

3) Superset computation

For each group the optimizer computes the superset of the individual results of the queries in that group. The motivation behind superset computation is taking care of common data. If the results of multiple queries have common data then this common data is present in the superset only once. So, if we broadcast this superset instead of sending the results individually we will have to transmit less data and thus save bandwidth. Let us consider some example query results below.

Table 3. Example Query Results

q1	q2	q3	q4
T1	T2	T3	T1
T2	T3	T4	T5
T3	T4	T5	T6
T7	T5	T6	T7
T8		T7	T8

The above table shows tuples comprising the result of queries q1, q2, q3, q4. If the query results were dispatched separately 19 tuples would have to be transmitted. But we note that there are only 8 distinct tuples. If we broadcast the superset Σ = {T1, T2, T3, T4, T5, T6, T7, T8} then only 8 tuples have to be broadcast.

Table 4. The Superset

T1
T2
T3
T4
T5
T6
T7
T8

The superset computation algorithm should perform better than brute-force superset computation algorithm. In brute-force method all the queries are evaluated separately and then their superset is computed. But the superset computation algorithm used in this work examines the predicates of the queries in the group and identifies the range of attribute values each query requires from each table in that group. In query Q1 we have the predicate "amt between 900 and 1400". From this we get the range [900, 1400]. Similarly, from predicate "amt between 950 and 1500" in Q3 we get the range [950, 1500]. The predicate "assets >= 1700000" in Q1 gives the range [1700000, MAX_ASSETS] and "assets >= 2100000" in Q3 gives [2100000, MAX_ASSETS]. Here MAX_ASSETS is the maximum value of the attribute assets in the table loan. The following algorithm is used for identifying the ranges. Here r_{ijk} is the data structure for storing the (range, query) pair for group i table j and attribute k, and table-list [i] gives the table list of group i.

The Algorithm :

```
for each group i
  for each table j in table-list[i]
    for each attribute k in table j
      for each query l in group i
        begin

          find the range R required by the query l
          add (R,l) to r_ijk

        end.
      end for {each query l}
    end for {each attribute k}
  end for {each table j}
end for {each group i}
```

The above algorithm gives the following results. r_{113} = { ([900, 1400], Q1), ([950, 1500], Q3) }, r_{123} = { ([1700000, 9000000], Q1), ([2100000, 9000000], Q3)}(Here MAX_ASSETS = 9000000). We see that [900, 1400] $\cup$ [950, 1500] = [900, 1500] and [1700000, 9000000] $\cup$ [2100000, 9000000] = [1700000, 9000000]. Now from the table loan we select tuples with 900 <= amt <= 1500 to from a new table loan' and from the table branch we select tuples with 1700000 <= assets <= 9000000 to form a new table branch'. From the data structure r_{ijk} we can find which tuple in these tables are needed by which queries. Now we join the tables loan' and branch' to get the superset.

Table 5. The table loan'

lno	bname	amt	Needed by
L-11	Round Hill	900	Q1
L-14	Downtown	1500	Q3
L-15	Perryridge	1500	Q3
L-16	Perryridge	1300	Q1, Q3
L-17	Downtown	1000	Q1, Q3

Table 6. The table branch'

bname	bcity	assets	Needed by
Brighton	Brooklyn	7.1M	Q1, Q3
Downtown	Brooklyn	9M	Q1, Q3
North Town	Rye	3.7M	Q1, Q3
Perryridge	Horseneck	1.7M	Q1
Redwood	Palo Alto	2.1M	Q1, Q3
Round Hill	Horseneck	8M	Q1, Q3

Now while performing the join operation we note that 3rd tuple (t1) in loan' joins with the 4th tuple (t2) in the branch' table. But t1 is needed by Q3 only whereas t2 is needed by Q1 only hence they are not joined. Otherwise the join operation will create a spurious tuple t1t2 that is not a part of any of the query's result. This technique eliminates spurious tuples. The 4th tuple in loan' is needed by Q1, Q3 and the 4th tuple in branch' is needed by Q1 hence they are joined. Two matching tuples t1 and t2 are joined if t1 (needed by) ∩ t2 (needed by) ≠ φ. After join operation we get the following join result.

Table 7. The join result

lno	bname	amt	bcity	assets	needed by
L-11	Round Hill	900	Horseneck	8M	Q1
L-14	Downtown	1500	Brooklyn	9M	Q3
L-16	Perryridge	1300	Horseneck	1.7M	Q1
L-17	Downtown	1000	Brooklyn	9M	Q1,Q3

The project operation gives the superset for group G1 as shown below.

Table 8. The superset to be broadcast

lno
L-11
L-14
L-16
L-17

Table 9. The indivigual result for Q1

lno
L-11
L-16
L-17

Table 10. The individual result for Q3

lno
L-14
L-17

The individual results for Q1 and Q3 are as shown in Table 9 and Table 10 respectively. The tuples in the superset are ordered in such a way that tuples belonging to the same query are as contiguous as possible (table 11). As a result the superset now consists of three blocks of BL1 = { L-11, L-16} , BL2 = {L-17} and BL3 = {L-14}.

Table 11. Superset with contiguous tuples

lno	Needed by
L-11	Q1
L-16	Q1
L-17	Q1, Q3
L-14	Q3

4) User specific encryption of the superset

We can see that in the above procedure the superset broadcast contains tuples that belong to multiple queries. Hence some tuples in the superset are not part of some queries. Each mobile device can read the whole superset but they accept some of the tuples and reject others according to the bit streams sent to them via point-to-point channels under their own volition. So there is every chance that a malicious user will accept tuples that are not part of his result because he cannot be forced to obey the bit stream sent to him. To solve this problem we have to encrypt the tuples belonging to a specific user by a randomly created key before broadcasting the superset. And these keys are sent to individual mobile devices via point-to-point channels. In the above procedure we already keep track of which broadcast tuples belong to which query. Our scheme is illustrated below.

Table 10. User specific encryption of the superset

Blocks of tuples	Corresponding users/mu's	Keys used for encryption	Sizes of encrypted blocks in units of time
BL1	U1	K1	S1
BL2	U1, U2	K2	S2
BL3	U2, U3	K3	S3
BL4	U3	K4	S4

Suppose the superset to be broadcast consists of 4 blocks of tuples BL1, BL2, BL3, and BL4. BL1 is required by user U1, BL2 by U1 and U2, BL3 by U2 and U3, BL4 by U3. The keys are randomly generated by the server for each bloack. BL1 is encrypted by K1, BL2 by K2 etc. The keys are sent to the mobile devices via point-to-point channels. The key K1 and K2 is sent to the user U1; the keys K2 and K3 are sent to U2; and

the keys K3 and K4 are sent to user U3. The server computes for each device the ranges of times during which that device has to tune to the broadcast channel. This can be computed by keeping track of sizes of different blocks of tuples. The server sends to each mobile device the start time and the end time of each range along with the keys. These informations are sent via secure point-to-point channels.

U1 receives {0,S1}K1 ; {S1,S1+S2}K2
U2 receives {S1,S1+S2}K2 ; {S1+S2,S1+S2+S3}K3
U3 receives {S1+S2,S1+S2+S3}K3 ; {S1+S2+S3, S1+S2+S3+S4}K4

5) Broadcasting the encrypted superset

The superset so computed is broadcast on the broadcast channel. Individual mobile devices have to find out individual results from the superset. The server also sends a bit stream to each mobile device through the point-to-point channels to them. This mechanism is explained in the next section.

6) Extraction of indivigual results

Each mobile device uses the information sent via point-to-point channels to tune to the broadcast channel for specific time periods and decrypt the information using corresponding keys and thus receive their own tuples.

B. Conclusion and scope for further work

A method of security in the context of query processing in mobile database has been presented in this paper. The proposed method can prevent unauthorized access of multiple query results sent through the same broadcast channel. Note that, in some previous work [10], no security measure has been used in the context of query processing in mobile database. Besides, they considered only subsumption while performing grouping of queries. So, with the query set Q1, Q2, Q3, Q4 only {Q2, Q4} will be grouped, but the queries Q1 and Q3 will be separately processed as there is no subsumption among them.

REFERENCES

[1] R. Malladi and D.P. Agrawal. "Wireless and Mobile Networks: Advances and Challenges." *Proceedings of the 4th World Multiconference on Systemics,* Cybernetics and *Informatics (SCI2000) and the 6th InternationalConference on Information Systems Analysis and Synthesis (ISAS2000),* , Orlando, FL, USA, July 2000, pp.218-223.

[2] T. Imielinski and B.R. Badrinath. "Data Management for Mobile Computing." *SIGMOD RECORD*, 1993, Vol. 22, No. 1, pp. 34-39.

[3] D. Barbara. "Mobile Computing and Databases – A Survey." *IEEE Transactions on Knowledge and Data Engineering*, Jan/Feb 1999, Vol. 11, No. 1, pp. 108-117.

[4] P. Sistla, O. Wolfson, S. Chamberlain and S. Dao. "Modeling and Querying Moving Objects." *Proceedings of the International Conference on Data Engineering*, Birmingham, UK, 1997, pp. 422-432.

[5] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. "Broadcast Disks: Data Management for Asymmetric Communication Environments." *Proceedings of ACM SIGMOD Conference*, San Jose, CA, May 1995.

[6] S. Acharya, M. Franklin, and S. Zdonik. "Balancing Push and Pull for Data Broadcast." *Proceedings of ACM SIGMOD Conference*, Phoenix, AZ, May 1997.

[7] F. Chen and M.H. Dunham. "Common Subexpression Processing in Multiple-Query Processing." *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 3, May/June 1998, pp. 493-499.

[8] M. Jarke. "Common Subexpression Isolation in Multiple Query Optimization." *Query Processing in Database Systems,* Springer Verlag, New York, 1985, pp. 191-205.

[9] T.K. Sellis. "Multiple-Query Optimization." *ACM Transactions on Database Systems*, Vol. 13, No. 1, Mar 1988, pp. 23-52.

[10] R. Malladi and Karen C. Davis. "Applying Multiple Query Optimization in Mobile Databases." Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), IEEE Computer Society, pp 294-303.

[11] T. Imielinski, S. Viswanathan and B.R. Badrinath. "Power Efficient Filtering of Data on the Air." *Proceedings of the EDBT Conference,* Cambridge, UK, March 1994.

D. Saha is with the Department of Computer Science and Engineering, Jadavpur University, Kolkata – 700032, India. Phone: 91 94330 65130; fax: 91 33 2413 1766; e-mail: neruda0101@ yahoo.com.

N. Chowdhury is with the Department of Computer Science and Engineering, Jadavpur University, Kolkata – 700032, India. e-mail: nirmalya_chowdhury@yahoo.com.