Techniques for Building Large Relational Databases on Mobile Computing Systems

Mukesh Singhal, Department of Computer Science
The University of Kentucky, Lexington, KY 40513

1 Introduction

Propelled by the availability of inexpensive portable computers (laptops, pocket PCs, personal digital assistants, etc.) and recent advances in communications technology, mobile computing represents a fast emerging technology [5]. A mobile computing system consists of a set of mobile computers (*Mobile Hosts*, MH) and fixed computers. The geographical area covered by the mobile computing system is divided into smaller regions, called *cells*. Each cell has a fixed computer, referred to as the *Mobile Base Station* (MBS). An MBS is connected to other fixed computers through a fixed wireline network and can communicate with MHs in its cell through wireless communication. A MH can communicate with other MHs in the network only through the MBS of its present cell. Through hand-offs and location management, a mobile computing system provides mobile hosts with continuous connectivity to the system, despite their random mobility.

Due to their cost-effectiveness, convenience, and the competitive edge they provide, mobile computing systems are gaining widespread popularity in several facets of day-to-day life such as healthcare, business organizations, crime prevention, and military. In particular, mobile information systems (which are databases on mobile computing systems) are getting increasingly popular as many applications require storage and online processing of huge information and as organizations strive to gain competitive advantages in selling products and services to increasing number of customers over the wireless networks.

However, the field of mobile information systems is in the state of infancy and much research work is needed to facilitate the implementation of databases on mobile computing systems. Databases in mobile computing environments are mostly used to support online applications and therefore, the performance is critical. Although a data model that is most appropriate for mobile information systems is a research topic, the use of relational databases is advocated due to their mathematical soundness, simplicity, and sheer dominance in the commercial market. Since join is the most expensive operation in query processing in relational databases, the main focus of this study is development of fast and low-cost join execution techniques that will work effectively in mobile environments. Traditional join methods are not suitable for databases on mobile computing systems because of their slow response and high processing cost. In addition, host mobility in mobile computing systems introduces many challenging issues [5] such as low bandwidth of wireless channels, limited computation power, small memory, and short battery life at mobile hosts, and hand-offs due to mobility, which render the existing relational database technology unsuitable to mobile information systems. Therefore, the biggest challenge a designer faces is how to take the query execution in relational databases which typically consumes substantial resources and is slow and make it work in an environment that is resource anemic and poses other problems.

Background: A number of join methods have been developed and perfected for use in commercial relational databases on static networks. A detailed discussion on joins methods can be found in [7]. All three major join methods, namely, nested loops, sort merge join, and hash based joins have been perfected to an extent that any further improvement in these methods will enhance the performance of join execution only marginally. Parallel joins on parallel databases have been investigated extensively. However, parallel join execution only provides a limited performance advantage since join executions are often I/O bound. The use of join index [2, 6] or pointer-based joins [4] to speed up join execution are also limited due to their huge storage overhead. The B_c tree approach [3] improves upon join index in that the additional data structure added for indexing purpose is much smaller than that in a join index approach. However, B_c trees cannot be used to speed up selection operations in non-join attributes and query results from one B_c or selection

cannot be fast propagated to another B_c. Without this capability, a full query execution, including multiple selections and joins, cannot be carried out entirely in B_c trees.

2 Technical Approach

Basic Concepts

The proposed database consists of two parts: the main database and the summary database. The summary database concisely extracts the information in the main database relevant for query processing. Query executions are first carried out in the summary database. This is a full fledged relational query execution in the sense that multiple selections and joins can be executed for a query in the summary database. (This is a novelty of our approach because no method has been proposed to do this earlier.) Query executions in the summary database are done very fast since the summary database is very small and these query executions give possible locations of the data of interests in the main database. After that, the results are mapped to the main database and query execution is done in the main database, which is very fast since the search in the summary database has already revealed possible locations of all data of interests in the main database.

The design challenges in implementing mobile information systems can be very well addressed by the proposed approach. The limited resources on MHs can be utilized by placing a small summary database on them. MHs can do the preprocessing of queries and pass the results to the main database (which resides on a large computer in the static network) for final query execution. The collective computing power and disk capacity of a large number of MHs constitute a significant portion of the overall system resource in a mobile information system and by judicially utilizing it, the user response time can be minimized and the overall system throughput can be greatly enhanced. This is especially true in times of heavy query workload because the higher the query workload, the more the active MHs that are available in the mobile information system. The problems resulting from host mobility will have a less effect on the overall system performance because a good part of a query execution can be done on an MH which is highly available (and fairly reliable) to the end user and doesn't suffer from all the complications brought in by the low-bandwidth and unreliable communication in wireless channels. In fact, a user can run several preliminary queries on its MH before phrasing the final query (based on the results of preliminary queries) and submitting it to the main database for the final execution.

Components of the Proposed Database

Summary databases: The summary database is comprised of summary relations. A summary relation is a complete database relation in the sense it has all attributes that its base relation (from which the summary information is extracted) has. For each join attribute in each relation of the main database, a summary relation is constructed that becomes a part of the summary database. A possible way to construct a summary relation so that it very concisely extracts information from its base relation, is as follows: A tuple in a summary relation corresponds to a group of tuples in its base relation and it is a tuple of signatures. A tuple in the summary relation contains group signatures at each attribute and these group signatures are obtained by superimposing individual signatures from the corresponding attribute in a group of tuples from the base relation. A summary database equipped with bonding tables and summary join indexes allows implementation of a full set of relational operations, including multiple selections and joins.

Bonding tables: Each summary relation contains group signatures, and the grouping is done based on values in join attributes of the base relation. Once summary relations are constructed, there is no way to establish connections between different summary relations of the same base relation by directly examing these summary relations. Bonding tables are constructed between each pair of summary relations of the same base relation to solve this problem. Different summary relations of the same base relation are connected by bonding tables. A bonding table is used to map results of selection or join operations from one summary relation to another summary relation that share the same base relation.

Every row of a bonding table has two components each contains a pointer of a tuple in one of the two summary relations. The tuples in summary relations pointed to by pointers in the same row of the bonding table correspond to some common tuples in the base relation.

Summary relations are much smaller than their base relations, and a base relation usually only has a few join attributes. Though it is conceivable that in the worst case, the number of rows in a bonding table could be the cross product of the numbers of tuples in the two summary relations that it connects, it will always be much smaller than the size of a join index [2]. In addition, the groupings that are used to build summary relations can be adjusted to reduce the size of a bonding table. In comparison, this flexibility does not exist in a join index.

Addressing tables: After searching the summary database, a set of tuples are found that are relevant to the query. These tuples represent possible locations of the data of interests in the main (base) database. A mechanism is needed to map tuples in the summary relations to corresponding tuples in base relations. A mechanism that works as follows (called the addressing table) can be used to provide such mapping: Logically, each entry in an addressing table has two components: the address of a tuple in a summary relation and a set of addresses of tuples in the base relation. The connection between addresses in these two components is such that the tuple in the summary relation contains group signatures at each attribute and these group signatures are obtained by superimposing individual signatures from the corresponding attribute in a group of tuples from the base relation.

An approach to implement addressing tables using B+ trees is as follows: Addresses of upper layer tuples can be on non-leaf nodes and addresses of tuples in base relations can be on leaf-nodes.

A join index, called the *summary join index*, will be used to connect two different summary relations that belong to two different base relations that are joinable. A summary join index will be employed to perform join between two different summary relations that stem from two different joinable base relations.

3 Conclusion

Given huge interest in relational databases in academia as well as in industries and explosive growth in mobile computing environments, it is inevitable that large relational databases will be deployed on mobile information systems. The low-cost and fast techniques for join operations developed in this project will be a significant step towards enabling and facilitating the deployment of relational database to mobile computing systems. The impact and the utility of the developed join techniques will go beyond mobile environments because these techniques can very well be applied to any large databases for performance improvement.

References

[1] B. C. Besai, P. Goyal, F. Sadri. A Data model for use with formatted and textual data. *J. Am. Soc. Inf. Sci. 37, 3 (May 1986)*, pp.158-165.

[2] P. ValDuriez. Join Indices. *ACM Transactions on Database Systems, Vol 12, No.2, pp. 218-246,*, June 1987.

[3] B. C. Desai. Performance of a Composite Attribute and Join Index. *IEEE Transactions on Software Engineering, Vol. 15, No. 2, 1989.*

[4] E. J. Shekita, M. J. Carey. A Performance Evaluation of Pointer-Based Joins. Proc. of 1990 ACM SIGMOD Int'l Conf. on Management of Data, 1990.

[5] G. Forman, J. Zahorjan, "The Challenges of Mobile Computing", *IEEE Computer*, April 1994, pp. 38-47.

[6] Z. Li, K. A. Ross. Fast Joins Using Join Indices. *Technical Report, CUCS-032-96, 1996, Columbia University.*

[7] Y. Yang and M. Singhal, A Comprehensive Survey of Join Techniques in Relational Databases, The Ohio State University Technical Report No. OSU-CISRC-10/97-TR48, 52 pages.