The Design of the Selection and Alignment Queries Using Mobile Program (J2ME) for Database Query Optimization

Cheon-Hong Min[a] , PrasannaKumar[b]

[a]Dept of Technical Management Information System
Woosong University Lecturer
17-2 Jayang Dong Dong Ku Daejeon 300-718
Tel : +82-42-630-9853, Fax : +82-42-630-9859, E-mail : minch22@chol.com

[b]Dept of Technical Management Information System
Woosong University Lecturer
17-2 Jayang Dong Dong Ku Daejeon 300-718
Tel : +82-42-630-9856, Fax : +82-42-630-9859, E-mail : pkumar@woosong.ac.kr

Abstract

In this paper, recognizing the importance of the database query optimization design methods, we implemented mobile database with mobile program (J2ME) which is a useful database procedures.
In doing so, we emphasize the logical query optimization which brings mobile database to performance improvement.
The research implies that the suggested mobile program (J2ME) would contribute to the realization of the efficient mobile database as the related technology develops in the future.

Keywords:

Database Query Optimization; Mobile Database; Mobile computing; Multi query approach

1. Introduction

The mobile database, or embedded database on a mobile device, is starting to become an important player in the industry. To make data management and sharing easier, there is a strong need for developing mobile database systems for mobile devices[5].
The embedded database on a mobile device is going to be the pioneer in the future mobile industry and it will challenge all the centralized database storage[7]. The query optimization of such databases will be more reactive than the traditional databases used for mobile devices. Many challenges to develop these small databases are being tried on portable computing device, including smart phones, PDAs (personal digital assistances), intelligent appliances and other embedded systems.
Miniature databases attached in mobile devices can interact through queries on a wireless communication. Omnipresent database attaches such DBMSs to real world "object" physically, and then allows different organizations to share information retrieved directly from physical goods, materials, or persons. We are trying to describe in this paper how to manage omnipresent database in the context of mobile commerce applications and how the J2ME based on mobile program can be used for the optimization database multi-query management in mobile devices.

The logical database design which transforms the real world entities to physical database designs has significantly affected the database system platform we want to be implemented properly. The logical database design process is able to determine desirable features reflecting organizational contexts.

Embedded system developers increasingly use Java, especially as microprocessors improve in their speed executing Java code. Sum Microsystems' J2ME, with its flexible user interfaces, robust security, and built-in networking, is proving especially popular for consumer electronics and mobile devices. Later, we will use J2ME to embody the possibility of mobile database although many types of mobile database products have been commercialized.

Particularly, the logical database design emphasizing user's perceptive has made differences in the performance of organizations with equivalent physical database resources.

There are many important issues about the logical database design, one of which is related to the optimization of database query. Database users have recognized that SQL programming for data manipulation is very important because it would make a difference in the organizational performance.

In particular, mobile database is highly costly in communication since it requires decentralization query strategies due to its mobility property[11]. Its implementation may need complicated methods so that we should optimize query strategies to meet the mobility property.

This research suggests an optimized query approach where the mobile program generates a database table, and a query program is designed that efficiently selects and aligns the records stored in the table.

In doing so, we emphasize the logical query optimization which brings mobile database to performance improvement.

The research implies that the suggested mobile program (J2ME) would contribute to the realization of the efficient mobile database as the related technology develops in the future.

The rest of paper is organized as follows : Theoretical Base is discussed section 2, In section 3, Design of Query program is described in details. Finally, we shall conclude the paper in section 4.

2. Theoretical Base(Concept of Mobile Database)

2.1 Efficiency of Database Query

Many databases support decision-making. Often this means choices between alternatives according to partly subjective or conflicting criteria. Database query language is generally designed for precise, logical specification of the data interest, and tend to be a awkward in the aforementioned circumstances.

When constructing the logical database design, the user, especially DBA (Database Administrator) needs a careful contemplation: he should be able to maximize efficiency relative to effort by making a design that reduces of cost database system operations. An inefficient database design would lead DBA to being blamed for low quality operations of database system[6].

The efficiency of database query is based on some criteria that how fast is the accurate retrieval of specified data and how its beneficial for the user query.

Object-oriented database system for Java relies on the language's reflection capability to inspect the object format at run-time and discovers the format of the stored object. But J2ME, which targets small-footprint and embedded system, omits reflection due to this function's complexity and resource requirements. This has limited the availability of OODBMSs for J2ME.

Most DBMS adhere to a widely accepted relational database model. A database model is a description of how data is organized in a database. In a relational database model, data is grouped into tables using a technique called normalization, which you'll find about later in this chapter.

Once a database and at least one table are created, a J2ME application can send SQL statements to the DBMS to perform the following: Save data, Retrieve data, Update data, Manipulate data and Delete data.

Let's consider a case where Table 1 consists of 10,000 records and Table 2, 1,000 records.

Retrieving records from Table 1 first, then from Table 2 and finally Table 3, as seen below in Figure 1 is a less efficient performance procedure than selecting reversely as in Figure 2. Actually, Figure 2, an appropriate database query is superior to Figure 1, an inappropriate query: Figure 2 needs at most 6 accesses, while Figure 1 needs at least 10,000 accesses.

We must seek for efficient record access methods that enable us to save time and cost.

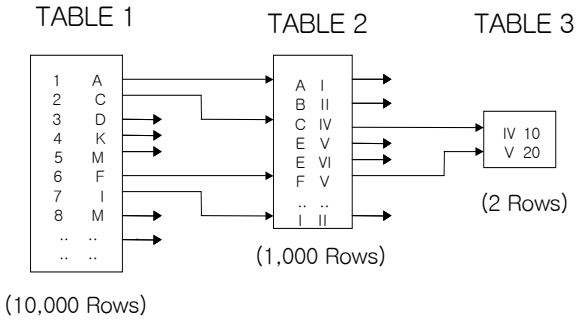


Figure 1 - Access Query from Tables : Case 1

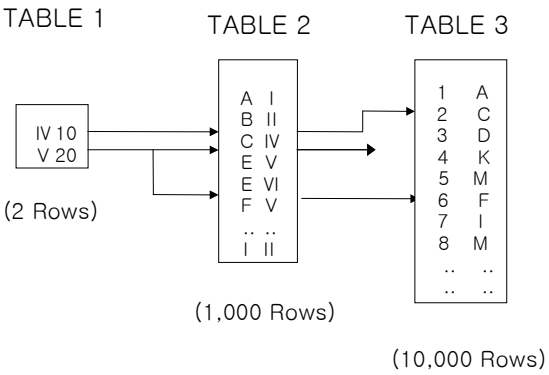


Figure 2 - Access Query from Tables : Case 2

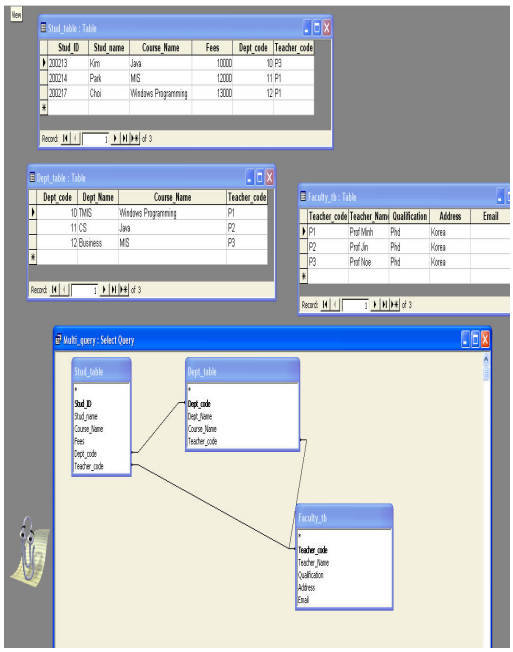


Figure 3 - Multi query

In the above, given multi query approach, we have to implement one method for retrieving required data from the multiple (three) tables. In the case of SQL server, it is an easy way that we can figure it out a joined query or sub query in the table structure. In the example, faculty table contains the primary key a teacher_code, which is the foreign key of the student table and the Dept_code, which is the primary key of Dept table, is the foreign key of student table.

So all the entry of the student table must undergo with the referential integrity with reference to the Faculty table and Dept table. In such case, there should not be one entry in the student table, which does not match with the teacher_code in faculty table or Dept_code in the Dept table.

So we can assign student table and Dept tables as the transaction table and faculty table as the master table. Keeping mind these criteria we have to take care the SQL when we create the all the three tables.

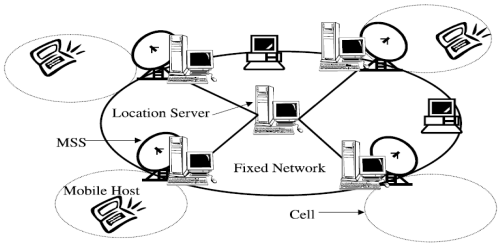
2.2 Query Optimization

Query processing in mobile database systems is significantly different from that in stationary systems and must hence be performed using different techniques[10].
Namely, the mobile system has to take into account limited resource like energy, communication bandwidth etc. And typical queries require to locate mobile users.
Query optimization is a common task performed by database administrators and application designers in order to tune the overall performance of the database system. Even if you have a powerful infrastructure, the performance can be significantly degraded by inefficient queries.
Therefore, query optimization has significantly influenced database performance in the strategic planning step for executing query. Thanks to query optimization, database SQL program allows us to use a high level query language beyond low-level data processing language.

2.3 Mobile Database

2.3.1 Mobile computing environment

As seen below in (Figure 4), mobile computer environment is composed of wire network and mobile hosts such as naptop computer, PDA, and mobile phone[8]. Mobile hosts communicate with the wire network computer, so called, mobile support station(MSSes). Each of mobile support stations manages mobile hosts to operate properly within a supporting geographical range, so called, a cell.



source : Data Management Issues in Mobile and Peer-to-peer Environments.2002
Figure 4 - Mobile Computing Environment

As a part of a mobile database system, a mobile host acts as a data client and a data server at the same time. A mobile hosts as a data server, is to support basic transaction operations such as read, write, commit and abort[2].
A fixed host called location server keeps information of location of all mobile hosts located within its management coverage. Location servers act as routers and are connected hierarchically by a high speed fixed network, where leaf nodes represent MSSes and non-leaf nodes represent location servers.

2.3.2 Mobility property and optimum locating

It is difficult to fix network addresses in mobile computing because of mobility property. Therefore, we have a difficult choice problem of selecting the best method to optimize locating the database that we wish to reach.
Much work has been conducted on moving objects, e.g., to keep track of mobile locations in telecommunication, transportation or traffic regulation applications.
It is said that it is still difficult for an optimizer to handle query processes even with a recent research results.

2.3.3 Query process in mobile database

A mobile computing environment directly affects the optimum strategy on mobile database query process. In general, mobile database query strategy is matched with decentralized query strategy because of mobility. Of course, in this case we should take into account communication costs resulting from decentralization.
In addition to costs of decentralization, the two features, mobility of mobile hosts and low integrity of wireless network, may make it difficult to adapt to mobile database the database model with fixed computer network[1]. Therefore, mobile database should have mobile transaction mode with more flexible structures.
We apply multiple query optimizations to batches of pull (on-demand) requests in a mobile database system. The resulting view can answer several queries at once, and it is broadcast on a view

channel dedicated to common answers of multiple queries rather than over individual downlink channels. A performance study is conducted that simulate different query workloads. The results indicate a significant saving in channel bandwidth usage and a reduction in average wait time for a multi-query approach compared to a traditional pull-based approach.

3. Design of Query Programs

DB Query is distributed client/server application for querying on databases with data protection against unauthorized access. It is portioned into two logical units, which run in conjunction on separate machines-thin client and remote server. In order to show query optimization with a mobile program, J2ME, especially selection query and alignment query out of the J2ME procedures, first we construct a "Student" table, secondly present SQL (Structured Query Language) program that is made according to relational database procedures.
Also, with mobile program (J2ME) we implement the process to which relational database is provided. That is, we reconstructed the "Student" Table. We implement selection and alignment query with mobile program (J2ME) methods.
Through this process, we would like to emphasize the embodiment on mobile database.

3.1 Database Table Design

First of all, we have to construct the student table in SQL Server as follows. Let's take an example of students in a university, who take courses at a particular point in time.
The data that belongs to a student are his ID, name, course name, term, advisor (professor) code, department code that he belongs to, course taken and the time of the course taken.
Suppose we are interested in searching students who meet a certain criterion.

create table stud_tb (stud_id int constraint pk_id primary key, stud_name varchar(30), course_name char(25),term char(3), Prof_code char(10), Dept_code char(10))

Similar way the all the other tables also have to be designed.

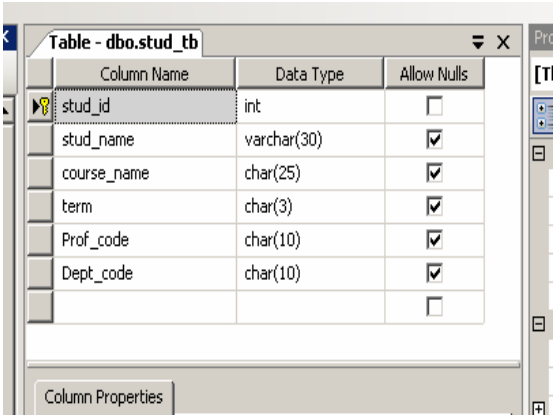


Figure 5 - student table

In general, database queries for special purpose are constructed using SQL as follows. The purpose of presenting the following queries is to show how to construct queries for the equivalent purpose in the mobile case.

3.1.1 Query searching from multiple databases

Database Query is distributed client/server application for querying on database with data protection against unauthorized access.
As in this case whenever we want to search query from multiple databases, we have to give emphases in data integrity, security and optimization methods.
A typical example is given below.
Join queries will be a better optimization method than the sub queries. So we implement join queries here.
select a.stud_id, a.stud_name, a.coursename, a.fees, b.dept_name, c.teacher_name from stud_tb as a join dept_tb as b, Faculty_tb as c on a.dept_code=b.dept_code and b.teacher_code=c.teacher_code

hence multiple join retrieve the queries.

3.1.2 Query searching for certain conditions

Conditioned queries rapidly process the results and save the cost and time.
For example, let's search for the students who took particular course.
select a.stud_id, a.stud_name, a.coursename, a.fees, b.dept_name, from stud_tb as a join
dept_tb asb on a.dept_code=b.dept_code where a.coursename='Java' or a.coursename='SQL'

3.1.3 Query aligning

For example, if we want to align students in the order of student id we can do it as follows.
select a.stud_id, a.stud_name, a.coursename, a.fees, b.dept_name, from stud_tb as a join
dept_tb as b on a.dept_code=b.dept_code order by a.stud_id

The alignment query output of this kind would help us to obtain valuable information such as interest, course royalty, and course-taking trends.

3.2 Design Queries Using Mobile Program (J2ME)

Sun's J2ME(Java 2 Micro Edition) wireless toolkit supports the development of java application that can be executed on mobile devices[10]. Therefore, J2ME is adopted to implement the mobile database.
Now, many types of mobile database is realized and has been served from many products. This paper focus on mobile program such as J2ME among the methods that is possible to construct mobile database.
Therefore, using J2ME, if we can show the queries of database, we suggest that mobile database be possible to embody. So, this paper is an example paper about constructing modern mobile database techniques.
This paper is emphasized that a forementioned query optimization of database can be possible by J2ME program such as table construction, search, selection and aliment.
As mentioned before, we would like to present how to construct queries in the mobile program(J2ME) for same purposes as in the general database case although many types of mobile database have been commercialized.

3.2.1 Database table design using "Class"

First, we want to store 6 fields from the "Student" table: stud_id(int), stud_name(var), course_name(char), term(char), Prof_code(char), Dept_code(char). To do it, we must construct "Class" by mobile program as follows.

```
//--
Public class Student {
        Private int stud_id ;
        Private var stud_name ;
        Private char course_name ;
        Private char term
        Private char Prof_code
        Private char Dept_code ;
        ...
        }
--//
```

3.2.2 Query searching the whole students (at random)

```
//--
public class GradeDB {
        RecordStore rs=null;
. . .
  public Vector retrieveALL() {
      RecordEnumeration re=null;
    Vector apps = new Vector();
     Try {
 re = rs.enumerateRecords(rf, rc, false);
    while(re.hasNextElement()) {
        int rec_id=re.nextRecordId();
     apps.addElement(new Appointment(rec_id, rs.getRecord(rec_id)));
}

} catch (Exception e)
  Finally {
 if(re!=null) re.destroy();
  }
return apps;
 }
. . .
}
--//
```

In the above program, retrieveAll() method uses RecordEnumeration to store all records. After we

filtered out all records with rs.get Record(rec_id) method, Byte array of each Student Record is divided by the field value.
When we have many records, searching all data at random requires squeezing all records, and thus a lot of cost and time.

3.2.3 Query searching the students who took course after particular time point

In this case, RecordFilter interface has only one method, Public Boolean matches(byte[] candidate) method, and is useful for using a criterion about record selection.

```
import javax.microedition.rms.*;

public class StudentFilter implements RecordFilter
    private long cutoff;
    public StudentFilter(long_cutoff) {
     cutoff=_cutoff;
 }

public boolean matches(byte[] candidate)
    Student app = new Student();
    app.init_app(candidate);

    if(app.getTime()>cutoff) {
      return true;
   }
  else {
    return false;
    }
 }
}
```

3.2.4 Query aligning students according to the time of courses taken

For this purpose, we can use RecordComparator interface as follows.

```
import javax.microedition.rms.*;
public class StudentComparator implements RecordComparator {
    public int compare(byte[] rec1, byte[] rec2) {
    Student app1 = new Student();
    app1.init_app(rec1);
    Student app2 = new Student();
    app2.init_app(rec2);
    if(app1.getTime()==app2.getTime()) {
     return RecordComparator.EQUIVALENT;
     }
  else if(app1.getTime()<app2.getTime()) {
     return RecordComparator.PRECEDES;
     }
Else {
Return RecordComparator.FOLLOWS;
}
}
}
```

Especially, RecordComparator interface could more easily align records in a table.
Compare(byte[] rec1, byte[] rec2) method is manipulated by three constants: PRECEDES, FOLLOWS, EQUIVALENT. If rec1 precedes (i.e., is smaller than) rec2, RecordComparator.PRECEDES is returned (i.e., produced). In the reverse case, RecordsComparator.FOLLOWS is returned. If rec1 equals to rec2, Records.EQUIVALENT is returned.

4. Conclusion

The database query optimization design has significantly influenced database performance. Therefore, the query optimization design issue has been studied continuously over time and is regarded as a critical issue.
The mobile database is expected to yield a great benefit in the time when we all want to access desired information at any time anywhere[3].
Recognizing the importance of the database query optimization design methods, we implemented mobile database with mobile program (J2ME) which is a useful database procedures. Using our suggested embodiment of mobile database, users are expected to query their desired information with a personalized device, such as naptop computer, PDA and mobile phone. If these mobile devices are connected to the database of a user's organization, he can make a decision on a timely basis. In achieving this objective, we should consider not only mobility of mobile devices but also low integrity of mobile transaction.
In short, we constructed a database table with mobile program (J2ME) and illustrated selection

and alignment query methods satisfying certain criteria. We hope these query methods of mobile program would enable mobile database to be utilized a lot more usefully and efficiently as database advances in the future. We must conduct continuous research on mobile database needs to meet personal desires.

References

[1] Back, H., Ku, K., and Kim, Y.(2001). "A Mobile Transaction Model Supporting Location-Dependent Query in Mobile Computing Environment," *Database Research*, Vol. 17 (3).

[2] Budiarto, S. N., and Tsukamoto, M.(2002). "Data Management Issues in Mobile and Peer-to-peer Environments," *Data & Knowledge Engineering* Vol 41. pp.183-204.

[3] Choi, M. and Kim, Y.(2001). "Mobile Database Summary and Research Trends," *Database Research*, Vol.17 (3).

[4] Helal, A., and Eich, M.(1995). "Supporting Mobile Transaction Processing in Database Systems," TR-CSE-95-003, Technical Report, University of Texas.

[5] Eric, J.L. L., and Yung-Yuan, C.(2004). "Design and Implementation of a Mobile Database for Java Phones," *Computer Standards & Interface,* Vol.26, pp.401-410.

[6] Jung, D.H.(1994). "A Study on the Implementation of Expert System for Database Performance Tuning," GSMIS Essay, Hankuk University of Foreign Studies.

[7] Kuramitsu, K., and Ken, S.(2001). "Towards Ubiquitous Database in Mobile Commerce," *Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access*.

[8] Lee, J.W.(2001). "mPowerAgent," *Database Research*, Vol.17(3).

[9] Lee, S,H.(2001). *Database system*, Jungik Publishing.

[10] Kottkamp, H.E., and Olaf, Z.(1998). "Location-Aware Query Processing in Mobile Database Systems," *Proceedings of the 1998 ACM symposium on Applied Computing*.

[11] Marsh, B., Douglis, F., and Caceres,R.(1993). "System Issues in Mobile Computing," TR-50-93, Technical Report, MITL.

[12] Yang, J.M..(2003). "Design and Implementation of a WAP Based Grade Inquiry System," GSEUCATION Essay, Hankuk University of Foreign Studies.

[13] Yu, F., and Jun, Z.(2001). *Wireless Java Programming with J2ME*, SAMS Publishing.