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

Ko, Wan Suk*, Min, Cheon Hong**

Contents

Keywords : Database Query Optimization, Mobile Database

Abstract

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.

* 한국외국어대학교 경영정보학과 교수

* Management Information Systems Professor, Hankuk University Of Foreign Studies, wsko@hufs.ac.kr, (02)961-4365

** 한국외국어대학교 대학원 경영정보학과 박사과정

** Management Information Systems Doctoral Program, Graduate school of Hankuk University of Foreign Studies, minch22@chol.com, 011-9475-2633

I. Introduction

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.

Especially, 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 programing 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 (Marsh, 1993). 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.

Ⅱ. Theoretical Base

1. Efficiency of Database Query

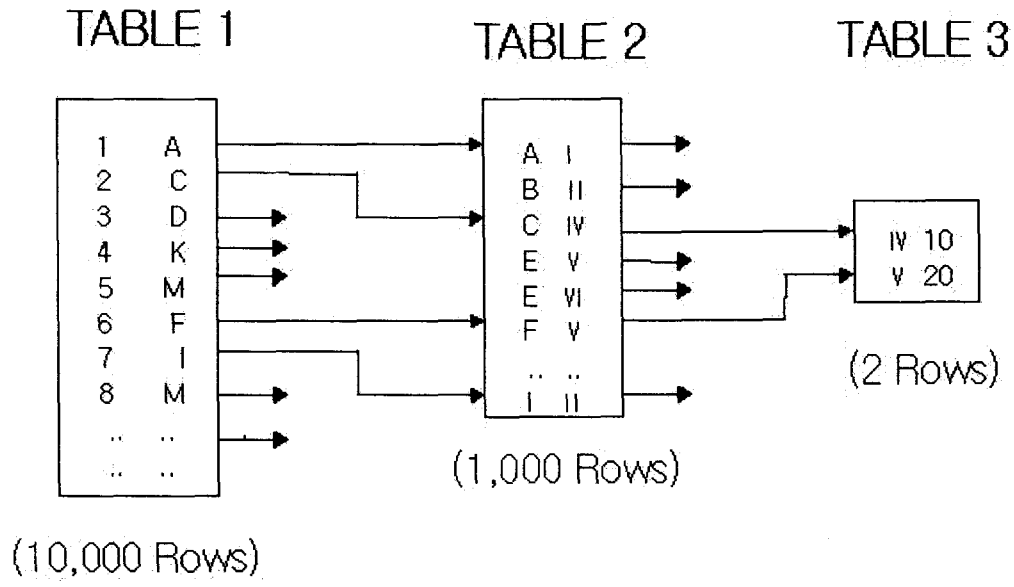
When constructing the logical database design, the user, especially DBA (Database Administrator) needs a careful contemplation: he should be able to maximize efficiency relative to effort by making a design that reduces of cost database system operations. An inefficient database design would lead

DBA to being blamed for low quality operations of database system (Jung, 1994).

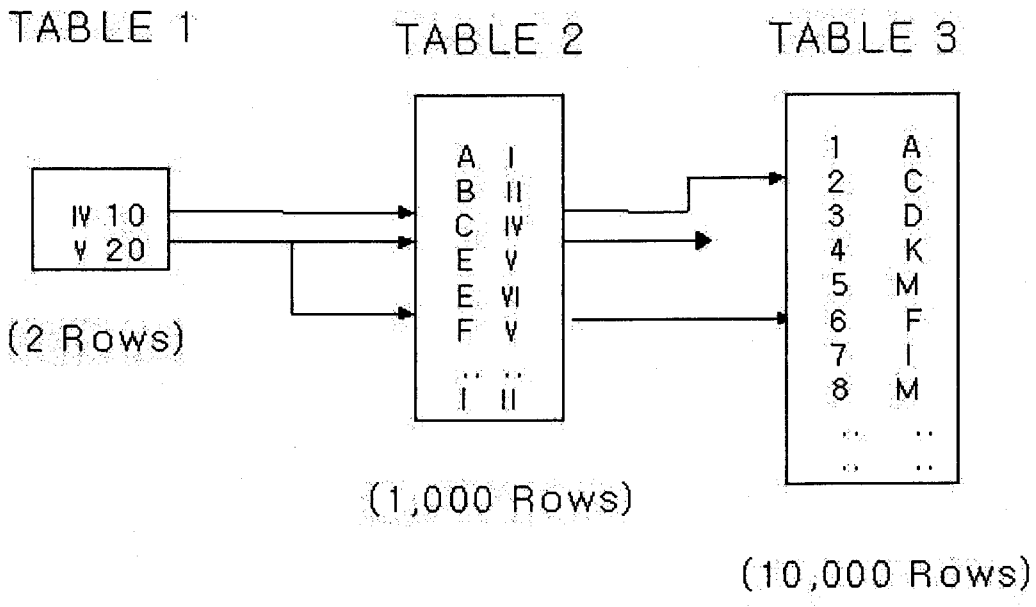
Let's consider a case where Table 1 consists of 10,000 records and Table 2, 1,000 records. Retrieving records from Table 1 first, then from Table 2 and finally Table 3, as seen below in Figure 1 is a less efficient performance procedure than selecting reversely as in Figure 2. Actually, 〈Figure 2〉, an 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



2. Query Optimization

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.

Now, we will implement a selection method and an alignment method out of

DBMS (Database Management System) procedures in order to demonstrate the capability of a high level query program. According to these logical methods, our operating manipulation about the order of calculations may have a significant effect on the query output in the alignment algorithm as seen below.

```
K=1 ;
    do i=1 to n ;
     do j=k to m ;
     if S(j)?A=R(i)?A then
          add R(i)?S(j) to result
        if S(j)?A > R(i)?A then
          leave inner loop ;
        end ;
        k = j ;
        end ;
```

3. Mobile Database

1) Mobile computing environment

As seen below in 〈Figure 3〉, mobile computer environment is composed of wire network and mobile hosts such as naptop computer, PDA, and mobile phone (Lee, 2001). Mobile hosts communicate with the wire network computer, so called, mobile support station. Each of mobile support stations manages mobile hosts to operate properly within a supporting geographical range, so called, a cell.

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.

It is said that it is still difficult for an optimizer to handle query processes even with a recent research results.

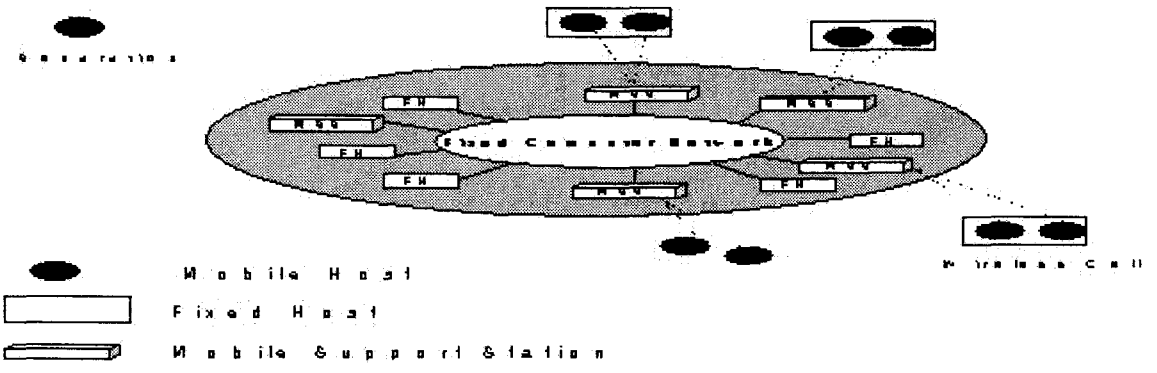
3) Query process in mobile database

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 (Back et al., 2001). Therefore, mobile database should have mobile transaction mode with more flexible structures.

In order to show query optimization with a mobile program, J2ME, especially selection query and alignment query out of

〈Figure 3〉 Mobile Computing Environment



* source : (Helal et al., 1995)

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.

Ⅲ. Design of Query Programs

1. Design Queries Using SQL

Let's take an example of students in a university, who take courses at a particular point in time.

The data that belong to a student are his ID, password, e-mail address, advisor (professor), department which he belongs to, courses taken and the time of the course taken.

Suppose we are interested in searching students who meet a certain criterion.

1) Database Table Design

Here, we construct "Student" table using SQL for the purpose of query designing. Of course, if we are interested in the student's grades, etc., we may need "Grade" table, "Professor" table, etc.

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.

2) Query searching the whole students (at random)

```
SELECT ID, PWD, EMAIL, PROF-CODE,
       DEPT-CODE, COURSE, TIME
FROM   Student
```

3) Query searching the students who took course after particular time point

In this case, we don't have to search the whole students in order to save time and cost.

〈Table 1〉 "Student" Table

Field name	Data type	Field length	Description
ID	Int	20	Student number (ID)
PWD	Var	10	Password
EMAIL	Var	100	Email
PROF-CODE	Var	10	Code assigned to the student's advisor
DEPT-CODE	Char	50	Department code
COURSE	Char	300	Course taken(NULL : No course taken)
TIME	Long		Time of the course taken

```
SELECT ID, PWD, EMAIL, PROF-CODE,
       DEPT-CODE, COURSE
FROM   Student
WHERE  TIME ) 12.10.3.10
```

(Note: 12.10.3.10 means 10 minutes after 3 o' clock, December 10th)

4) Query aligning students according to the time of courses taken

We can align students on the basis of the time of courses taken. The alignment query output of this kind would help us to obtain valuable information such as course interest, course royalty, and course-taking trends.

```
SELECT ID, PWD, EMAIL, MASTER_CODE,
       SECTION_CODE, SUBJECT
FROM   Student
ORDER BY TIME
```

2. Design Queries Using Mobile Program (J2ME)

As mentioned before, we would like to present how to construct queries in the mobile case for same purposes as in the general database case.

1) Database table design using "Class"

First, we want to store 6 fields from the " Student" table: ID(int), PWD(var), EMAIL(var),DEPT-CODE(String), COURSE(String) and Time(long). To do it, we must construct "Class" by mobile program (Feng yu and Zhu zun, 2001) as follows.

```
//—
Public class Student{
        Private int ID ;
        Private var PWD ;
        Private var EMAIL ;
        Private String DEPT-CODE ;
```

```
        Private String COURSE ;
        Private long TIME ;
        ...
        }
—//
```

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) 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;
    }
  }
 }
 —//
```

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;
     }
   }
  }
  —//
```

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.EQUVALENT is returned.

Ⅳ. 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 (Choi et al, 2001).

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 (Back et al, 2001).

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, hyung, Ku kyung, Kim yu. "A Mobile Transaction Model Supporting Locatio-Dependent Query in Mobile Computing Environment," *Database Research*, 2001; vol.17 (3).

2. Choi, mi and Kim young. "Mobile Database Summary and Research Trends," *Database Research*, 2001; vol.17 (3).

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

4. Jung, duek hyun. " A Study on the Implementation of Expert System for Database Performance Tuning," HUFS GSMIS Thesis, 1994.

5. Lee, jae woo, "mPowerAgent," *Database Research*, 2001; vol.17 (3).

6. Lee, suk ho, *Database system*, Seoul: Jungik Publishing, 2001.

7. Marsh, B., F. Douglis, and R. Caceres, "System Issues in Mobile Computing," *Techical Report TR-50-93*, MITL, 1993.

8. Yang, jung mi. "Design and Implementation of a WAP Based Grade Inquiry System," HUFS GSEUCATION Thesis, 2003.

9. Yu, Feng and Jun Zhu, *Wireless Java Programming with J2ME*, Seoul; Sams Publishig, 2001.