Advanced Data Organization for Java-Powered Mobile Devices

Tomáš Tureček, Petr Šaloun

Advanced Data Organization for Java-Powered Mobile Devices

Tomáš Tureček[1], Petr Šaloun[2]

[1] VŠB TU Ostrava, 17. listopadu 15, Ostrava, 708 00, Czech Republic, tomas.turecek@vsb.cz
[2] Ostravská univerzita, Dvořákova 7, 701 03, Ostrava, Czech Republic, petr.saloun@osu.cz

Abstract: The paper reports on actual research motivated by need for efficient data storage on J2ME CDLC 1.0+ MIDP 1.0+ platform. Presented solution fulfils those needs by providing Midletbase package implementing a relational database with user friendly and extensible interface.

Keywords: Java, J2ME, Database, Midlet, Midletbase, storage.

1 Introduction

Mobile devices are all around of us. They became so cheap and so useful that almost everyone of us has its own cellular phone. Today's cellular phones are quite universal as they can run various applications. The paper reports on continuation of previous research related to remote access to information systems [2] and it describes advantages of using Java-powered portable devices and wireless connections. The paper presents research on J2ME database framework we are working on.

1.1 Motivation

There are many ways of integrating portable device with existing systems. One is to extend an existing IS with some interface and to implement thick client application into the mobile device communicating with the IS through this interface. We have applied this approach as it allows implementation of more complex logic into thick client. That is an advantage in cases like offline access to IS data. During our work on synchronization of thick IS clients we have noticed some weaknesses of J2ME storage capabilities. We were forced every time to create custom wrappers around RMS storage (Record Management System [5]). Those problems have inspired us to create extension to RMS in form of relational database.

1.2 Problem to solve

Thick client application demands some kind of storage to persist needed data. Reason for that can be fount in [1]. J2ME offers just simple storage called RMS (Record

Management System [5]), which is in fact capable of storing byte arrays only. Developers are forced then to invent again and again their own RMS extension to persist primitive Java types. We have focused on this problem and we have created RMS extension simulating relational database over simple RMS storage.

2 Solution

State of the art was quite empty in the time we have started with our own database implementation. Only available solution was commercial solution PointBase Micro [4] and series of implementations of remote access to remote databases like Oracle Database Lite Edition [6]. These particular solutions did not satisfy our needs because we needed combination of all those solutions. We needed J2ME relational database which can be easily extended with new functionality such as remote access to remote systems.

Presented solution behaves as usual relational database but it uses RMS for storing data. We call the extension as Midletbase (Midlet [5] database). Extension offers easy-to-use API to database engine allowing creating or dropping tables and inserting/ updating/deleting/searching table entries. Our database framework can look like another J2ME layer to J2ME developers (see Figure 1).

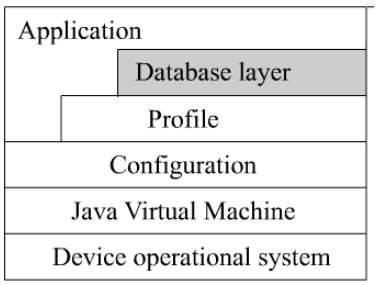

Fig. 1. Midletbase represents additional (database) layer to J2ME

Midletbase adds to J2ME following features:

- Data to store are organized into entries with defined data types.
- Entries are stored in tables.
- Tables are organized in databases.
- Database structure as well as data is persisted in RMS to have it accessible even after device restart.
- Data is accessible through simple API with the same way that it is usual in relational databases.
- API allows creating and dropping tables.
- API allows inserting, updating, deleting and searching over data in RMS.
- Midletbase is extensible so it enables developer to extend Midletbase functionality *plug-in* way and implement his/her own behaviour for database operations.

Preliminary implementation results were already published in [1]. Inspiration for our solution was based on our experience with J2ME and also on existing solutions [4, 6]. As the target platform we have selected CLDC 1.0 and MIDP 1.0. Reason for that is the Midletbase framework then will be usable on every single cell phone with Java support.

2.1 API

One of the requirements for the Midletbase framework was usability. Interface allows developers to use the storage with a simple way and following piece of code shows how easy is to initialize storage and to create table and insert there some entries.

Example of storage initialization and table interface usage:

```
// initializing storage
Storage storage = Storage.getInstance();
// creating table
ITable table = storage.createTable(
  Dbname,          // database name
  tblname,         // table name
  new Column[]{    // table structure
    new Column(
      id,                  // column name
      IOperator.INTEGER,   // column type
      false,               // can be null?
      true),               // is the key?
    new Column(name, IOperator.VARCHAR, false, false)
});

// adding entry to table
table.addEntry(
  new String[]{id, name},        // columns for values
  new Value[]{                   // values to insert
    new Value(10),               // Value class ensures
    new Value(some string)});    //       type safety
// searching in table
IResultTable result = table.getEntry(
    new String[]{id, name},        // result columns
    new String[]{id},              // search columns
    new Value[]{new Value(10)},    // search values
    new int[]{IOperator.EQUALS});  // search operators
// printing result vector to console
System.out.println(result.getAllEntries());
```

2.5 Extensibility

Midletbase database engine implements *Interceptor design pattern* [3]. Database engine is implemented as a chain of Interceptors and developer (using Midletbase) can create his/her own interceptors and plug them into database engine and to create this way some additional functionality (e.g. logging or triggers to his/her application). Developer is even able to completely change the engine behaviour like not to store data to RMS but to create connection to the server application and store data there. Only things needed for the change is to implement simple well-described `IStorageInterceptor` interface and to use different constructor for `Storage` class initialization.

Example of storage initialization using custom interceptors.

```
// initializing storage
Storage storage = Storage.getInstance(
    new String[]{
      mypackage.MyInterceptor1,
      mypackage.MyInterceptor2});
```

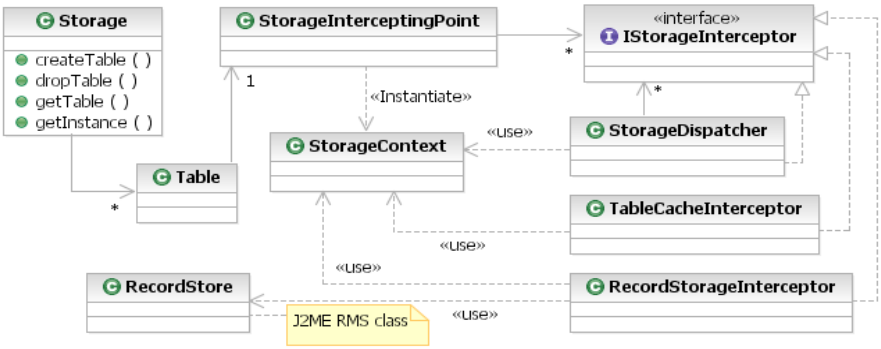

Fig. 2. Simplified class diagram depicting interceptor design pattern involvement in architecture and how the custom interceptors can be passed from `Storage` initialization through `Table` to `StorageInterceptingPoint` class.

2.5 Measures

We have performed set of measures showing characteristics of algorithms used in the implementation. They can be used for estimation of the effectiveness of Midletbase usage for certain purposes. This section shows characteristics of two usually used Midletbase use cases – create table and search the table. Following charts show average results from 20 measures. Unfortunately the short space in this paper prevents us from showing more characteristics of the framework.

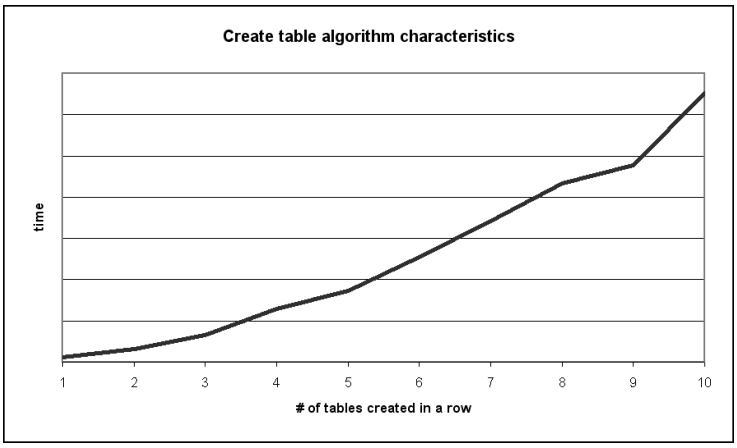

Fig. 3. Characteristics of the create table algorithm.

Figure 3 shows the create table algorithm characteristics. Dependency is not linear and it opens opportunity for future work to optimize algorithms for better performance. Objective of the measure is to get chart trend so the concrete time values are not relevant here; also the measuring is done with emulator[1].

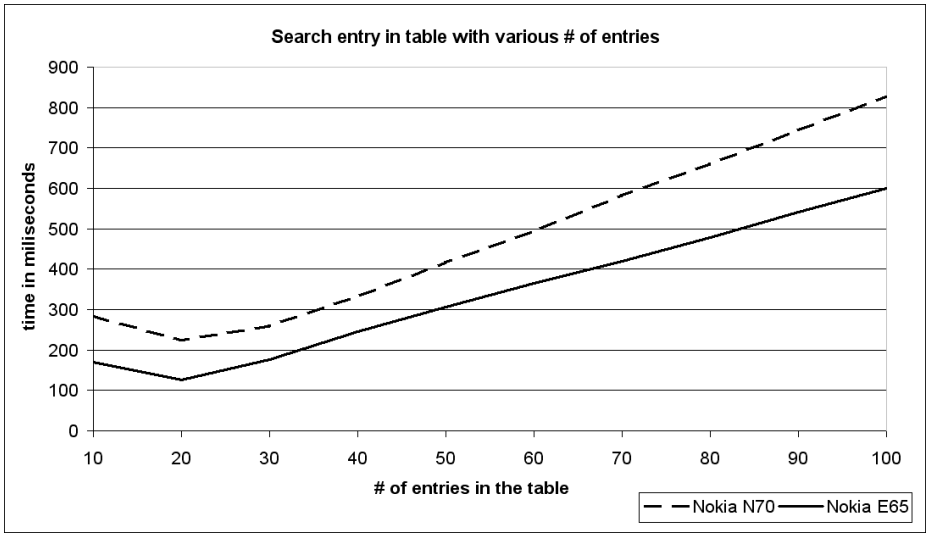

Fig. 4. Characteristics of the search table algorithm.

Figure 4 shows needed time for searching for entries (by key) in table with different number of entries. The characteristic is quite linear with some small overhead in case of first record store access. Chart above represents search without cache.

[1] Performed with Sun Microsystems Wireless Toolkit emulator 2.5.3 on Dell Latitude D620 laptop (CPU Genuine Intel T2400, 2GB RAM)

3 Future work

We are now working on JDBC driver for Midletbase to allow developers to use also standard DB interface.

Midletbase does not contain functionality related to database structure change. Tables and databases can be created or dropped but not altered. Functionality is not included so far because it is not needed functionality when we consider the framework usage – storage for deployed application. Such application has already steady DB structure.

In future we want also to focus on algorithms used in Midletbase. As we can see from section 2.5 there are opportunities to optimize those algorithms.

The topic of the research is really actual and solves incompleteness of today used standards. As a proof we can take [7] where we can find similar solution to our independent one but our solution is more focused on extensibility and usage.

4 Conclusion

The paper presents an ongoing research responding to real needs reported by many developers of Java mobile applications. The need is to have smart-enough storage on J2ME CDLC 1.0+ MIDP 1.0+ platform. Presented solution fulfils those needs by providing a package implementing a relational database with many features and a user friendly interface. The work is still ongoing but we plan to release Midletbase package as soon as we get a stable well tested version. The current version is available only for pilot applications.

References

1. Tureček T., Běhálek M.: Data organization on Java powered mobile devices, proceeding Datakon 2005, Brno, 2005, ISBN 8021038136

2. Tureček, T., Beneš M.: Remote Access to Information System via PDA, ISIM 2003, Brno, 2003, 145-152, ISBN 80-85988-84-4

3. Interceptor design pattern on Daily development [Online, 2008-07]. Available at: http://dailydevelopment.blogspot.com/2007/04/interceptor-design-pattern.html

4. PointBase Micro [Online, 2008-07]. Available at: http://www.pointbase.com/

5. Keogh, J. J2ME: The Complete Reference, McGraw-Hill/Osborne (2003), Berkeley.

6. Oracle Database Lite Edition. [Online, 2008-04]. Available at: http://www.oracle.com/technology/products/lite/index.html

7. Alier M., Casado P., Casany M.J.: J2MEMicroDB an open source distributed database engine for mobile applications, PPPJ '07, 2007