A Novel Storage Embedded Application

JINGYU ZHANG, RAFAEL CALVO, CRAIG JIN, RICHARD SHEPHARD
School of Electrical and Information Engineering, School of Electrical and Information Engineering, School of Electrical and Information Engineering, Australian Biosecurity CRC
The University of Sydney NSW, 2006
AUSTRALIA

Abstract: - This paper presents a solution for embedded database application architecture. Many mobile applications contain enormous data and intricate search which are easy to management if those data are preserved into one hard disk or the Disk Array. Therefore, organizing data structure according to device's capacity and system's priority is the way to improve system usability, availability. In order to enable handling enormous information, we propose a database application architecture, which provides reliable and configurable data storage services to embedded application system with the performance, scalability and reliability.

Key-Words: - Embedded Database Application, Architecture, and Software Design.

1. Introduction

With the rapid growth in mobile technologies and the inadequate coverage of wireless networks, the demand of available embedded applications is becoming ever greater; meanwhile, the computing landscape is much richer and more diverse than before. To collect information's organization in a way that a computer program can quickly obtains desired pieces of data are becoming the standard feature for most mobility applications.

One solution of database functions is that using wireless channels receive information from database server, which so-called mobile computing paradigm extends [1]. Dealing with Massive data, some researches thought that put all data on the Internet [2]. With network availability, mobile users access personal data by a Data Server Provider (DSP)[3]. However in some cases, they may have no networks coverage at all or wireless networks are vulnerable to frequent disconnection. So a new type of database management system (DBMS), the In-Memory Database Systems adapt to mobile systems which load all data to RAM from files when application start up, which is installed in set-top boxes, network switches and consumer electronics[4, 5]. The database APIs are used to the application requests the data items and all the operations to database cache are totally written to the files. From the performance aspect, it is a good approach to use In-Memory Database because it faster overall responsiveness than a traditional DBMS[6]. In addition, In-memory databases are less complex because it has fewer moving parts or interacting processes.

In generally, however, today's smart phones and PDA own 128MB of flash and 64MB of RAM. But, a phone OS need occupy many of available storage[7]. As a result, every byte used by the Embedded OS and foundational application such as middleware is a byte not available to developer for additional features. For example, O2 XDA Atom (http://www.seeo2.com/) Installed RAM for program is 50.39MB. To apart from the memory that used by OS and essential applications, totally, 14MB is free for users' applications. Worst of all, the free memory is not entirely used by one application because the Operating System reserves several memories for logs or backup function which mainly focus on Disaster Recovery. Meanwhile, Operating System assumes not only one application run on the device, so some memory is reservation for other application. Thus, the rest of memory for one application may less than 7 or 8 MB. It is clearly that the majority of database applications overstep the limit of devices providing. Normally, the size of database has been reduced significantly to match the memory[8, 9]. Whereas, decreasing database will results in losing functions which may not match the requirements of customers.

In this paper, we describe the architecture of mobile database application. Section 2 presents some previous work in structuring mobile applications. Section 3 presents conclusions.

2. System Anatomy

First, the mobility architecture will be described. Then, there are some in-depth descriptions of the data search and storage structures. Finally, the evaluation of performance will be scale.

2.1 Architecture Overview

In this section, we will give a high level overview of how the whole system structure looks like pictured in Figure 1. The framework consists of three layers: an Impute/Output tier, a Procedure Logics tier and a Data Access tier.

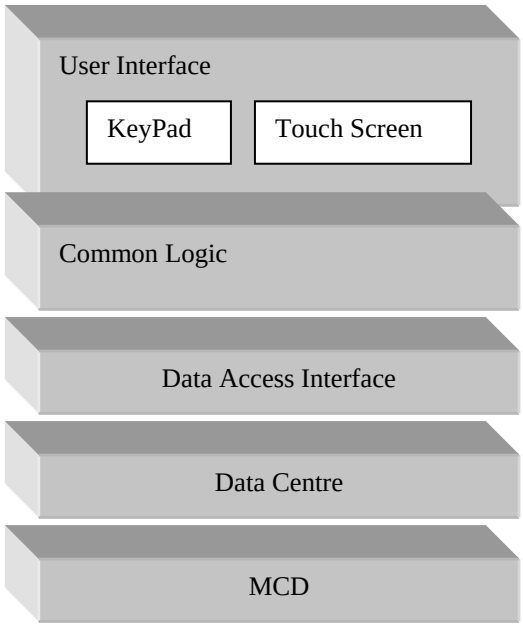

Figure 1 Archetecture

Firstly, the Impute/Output tier is the aggregate of methods by which users interact with a particular system, device. It handles presentation and interfaces with the operating system, the network and other software tools. It provides two functions, input and output. For the whole system, it is extremely important for the mobile database application because there are not any screen's standards for the mobile phones or handhold devices. To inputting information, people may use keypad; touch screen or voice inputting to let system accept information. To outputting information, the dimensions of mobile monitors are great deal different from one manufactory to another.

The Procedure Logics Tier covers the core business model which provides the basis for all business transactions which manages the business operation of the application. Additionally, The Procedure Logics Tier determines the behavior of the application in response to different situations, such as the need for validation and approval of final products before reporting. Moreover, this Tier includes the organization's workflows and operating procedures. Hence, it comprises the most expensive component in the implementation process.

Finally, the Data Access tier provides a Data Access Interface (DAI), a Data Centre (DC), and Memory Card Database (MCD). The DAI duty is as much as shielding by which the database operations are hidden from other parts. As the purpose of the architecture is that operating enough data to stand-alone mobile database, the role of DAI is smoother operation of the mobile platform for retrieving data and easer shifts the structure of database organization. In addition, the DAI quickly transfer users' intents to database acceptable commands. For the architecture, one of key components is the DC which is a platform-independent Java database and all the features have been loaded into the program memory when system start-up. Meanwhile, it provides effective data management for rapid and efficient mobile enterprise applications. Compared to traditional databases, DC eliminates disk I/O and provides Extra-SQL. Another key component is the MCD. MCD is a multi-attribute indexing file system, which is saved in Storage Memory.

In the Multiple-layers Application, all components are contained in greatly independent and separate tiers. This separation of User Interface, Procedure Logics, and Database Tiers which allows each Component to be used, replaced or reused, in new combinations that meets specific and dynamic business requirements[10].

2.1.1 Data Access (DA) Tier structures

In this tier, all of the database and files retrieve and store functions are encapsulated in this component. Each Object Oriented object has a shield around it and objects can not communication with each other. The way that they exchange information is though Data Access Tier in which all the database API are interconnected through the Data Access Tier. This component keeps the systems flexible. The business process can change easily. Additionally, the customer does not care about the format of database in which may come from DC or MC.

Object Oriented experts can build DA Tier as a flexible system because every module of the system can change independently, no impact to the other modules. Features' reuse has become a strongpoint in this architecture due to its potential benefits, which include increased product quality and decreased product cost.

2.1.2 Data center

The architecture of the Data Centre is shown in Figure 2. Data Centre is the key components of Data Storage

Layer by which data cache and in-memory database are combined together. It handles the cache and storage of Message, Transaction Data, and User Data. The Authentication Manager may be considered as optional function for stand-alone mobile database system because most of embedded devices support it. Thus, for improve performance, this part can as an additional choice.

Query Parser is the parts which analyze syntactic units which are originate by users in order to obtain the query results from database. The parser is a function that carries out the task which converts input text into a data structure, which is suitable for later processing and which captures the implied hierarchy of the input.

The database contains the tables and Recovery Log. To Recovery Log, it is a sequence of log records, which include transaction identifier, the data item written. Database modifications can be record in term of Recovery Log.

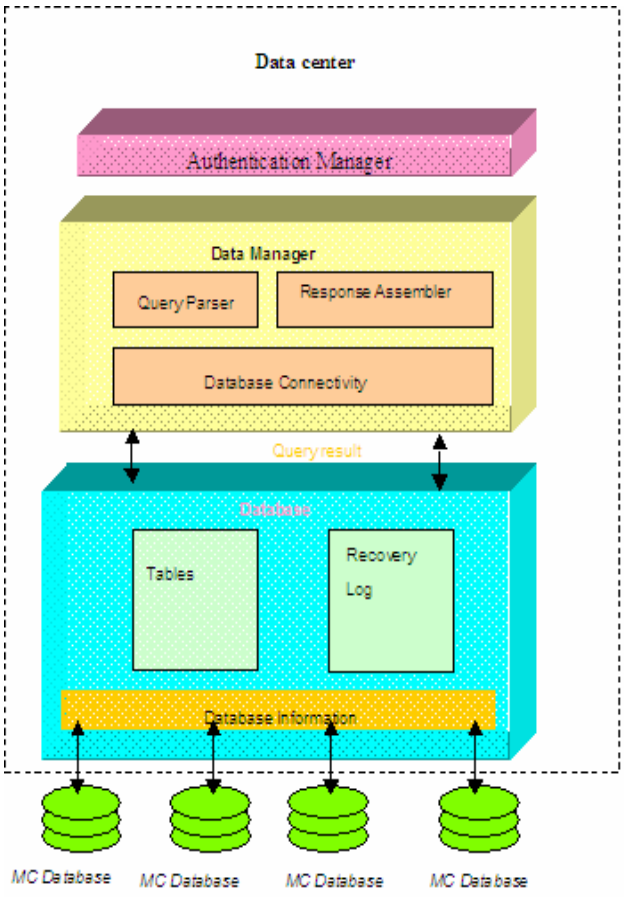

Figure 2 Data Center

There are many types of database structures can be use for DC such as Relational Database structure, Object-Oriented Database structure, and Object-Relational Database structure. Since the benefits of Relational Database overweight others, the DC's functions are described as a relational database. First of all, the key benefit of a Relational Database is that it can deal with complex relationships between objects. One reason of people use enormous data with mobile device is that the data contain many objects and the relationship around those objects are really involutedly. Secondly, a relational database was selected because it is used for storing large amount of data. The data organization for the relay settings data was arrived at through a process. Therefore, to enter, organize and select information in table of rows and columns, Relational Database is the ideal method. Moreover, in certain area, a relational database is a rational solution, especially in terms of analysis. The databases let data is manipulated in complex, interesting ways, allowing you to retrieve all records that match users' specific requirements, cross-reference different tables, and update records in bulk.

The principle of creating a database for embedded systems is determined by the application's requirements. The structures and features vary widely from vendor to vendor because phones are not built to a standard architecture. For instance, different manufactories define the concept "embedded systems" and "embedded database" to mean different things. Thus, the better way to save voluminous user data is to create individual data structures base on MCD, and then integrate that with your application.

To implement the database, we create the HB-Nary tree structure. The HB-NAry tree is a multi-attribute search structure with behavior similar with the single-attribute N-Ary tree [11] with holey brick function. Using holey brick (HB) deal with smaller bricks removed in k-dimensional space. To the objective of saving space, the empty nodes do not be contented in HB-Nary tree. Hence, the number of nodes in the tree is $\frac{N^h - 1}{N - 1} - \Delta d$, if the heights are h and empty nodes are Δd .

$$\sum_{i=1}^{h} N^{i-1} - \Delta d = N^0 + N^1 + N^2 + \cdots + N^{h-1} - \Delta d = \frac{N^h - 1}{N - 1} - \Delta d$$

Figure 3 represents the storage components of MCD. Indices, Data dictionary and Data files are included in MCD. The Indices provide fast access to data items that keep particular values. The Data files are located in the tree's nodes which are saved by encrypt file format. In addition, the structure of the database is describes by Data dictionaries which are used

frequently. Therefore the database system split the Data dictionary into two parts in term of priorities.

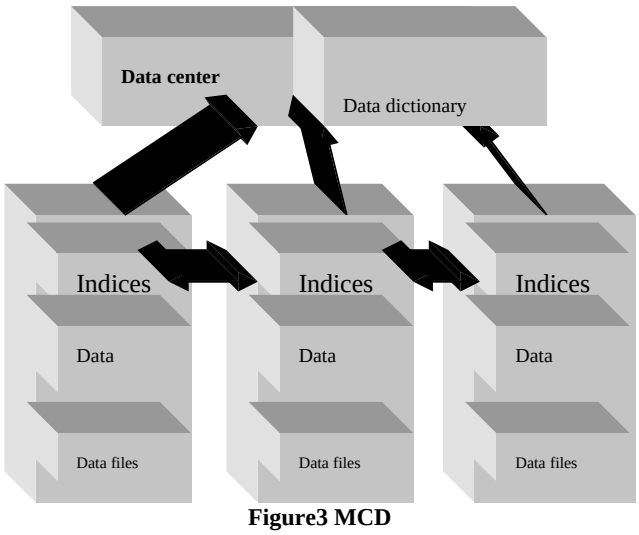

Figure3 MCD

Database Indices

The database index keeps information about each document. The index structure is associated with a particular search key such as ordered by fileID which is considered as the keys in sorted order, and associates with each search key. The information stored in each entry includes the current status, a file pointer. Forward Index and Inverted Index are defined in the indices.

Data dictionary

The dictionary has several different forms. One important change from other systems is that the lexicon fit in DC for usability and performance. In the database management system, the dictionary defines the basic organization of a database, which contains a list of all files in the database, the number of records in each file, and the names and types of each field[12]. Without a data dictionary, a database management system cannot access data from the database.

3. Conclusion

This paper addresses a novel mobility stand-alone database structure in an embedded real-time system. In particular, the architecture is suitable for gigantic data searching and storage application. According to our experimental evaluation, this architecture is accurate and effective enough to give the results optimized.

The basic idea for the architecture is to improve the availability and usability for embedded database applications. To realize this idea, we proposed a DC and MCD structure to locate data into different parts of embedded device and organized them in term of HB-NAry tree. As a result, the architecture can be regarded as an effective solution to embedded database application and it has already been running on the Bovine Syndrome Surveillance System (BOSSS)[7] which is syndrome surveillance systems for cattle.

References:

[1] A. S. X. Hong V. Leong, "On adaptive caching in mobile databases," in *ACM symposium on Applied computing*: ACM, 1997.

[2] H. H. G. C. D. Y. L. Xiaoming, "Global file store: a massive storage system on the internet concept and design," in *IEEE International Conference on Computer Networks and Mobile Computing,* : IEEE 2001.

[3] C. Zuji Mao; Douligeris, "A distributed database architecture for global roaming in next-generation mobile networks," *IEEE/ACM Transactions on Networking,* vol. 12, pp. 146-160, Feb. 2004.

[4] M. A. Olson, "Selecting and implementing an embedded database system," *Embedded Systems,* vol. 33, pp. 27 - 34 Sep 2000

[5] N. Wyatt, "Pure Java databases for deployed applications," in *16th IEEE International Conference on Data Engineering*, 2000.

[6] C. K.-L. L. W. M. Z. W. R. Yu, N., "A methodology to retrieve text documents from multiple databases," *IEEE Transactions on Knowledge and Data Engineering,* vol. 14, pp. 1347-1361, Nov.-Dec. 2002.

[7] J. Zhang, Calvo, RA, Shephard, RW & Jin, C, "A Framework for Mobile Disease Report and Investigation. International Conference on Mobile Technology Applications and Systems," in *3rd International Conference on Mobile Technology Applications and Systems* Bangkok, 2006.

[8] C. S. K. Thomas J. Marlowe, James W. Benham, "Design patterns for database pedagogy: a proposal," *ACM SIGCSE Bulletin,* vol. 37, pp. 48-52, February 2005.

[9] F. H. De Marchi, M.-S.; Petit, J.-M., "Some remarks on self-tuning logical database design," in *Data Engineering Workshops, 2005. 21st International Conference on*: IEEE, 2005.

[10] L. B. H. Peter Honeyman, "Extending the usability of mobile computers Communications and Consistency in Mobile File Systems," *IEEE Personal Communications,* vol. 2, pp. 44-48, December 1995 1995.

[11] E. a. B. Salzberg, "Using the Holey Brick Tree for Spatial Data in General Purpose DBMSs," *IEEE Database Engineering Bulletin,* vol. 16, pp. 34--39, September 1993 1993.

[12] M. A. Swain, J.A.; Korrapati, R.; Swain, N.K., "Database programming using Java," in *SoutheastCon, 2002. Proceedings IEEE,* 2002.