

University of Missouri-Columbia

Department of Electrical and Computer Engineering

ECE 7220 Real Time Embedded Systems

Fall, 2013

Project

# Remote Information Query System based on Faircom Database

Wenqiang Bo

December 13th, 2013

## **Abstract**

The project described in this report is a remote information query system (RIQS). Unlike other remote information query systems, my RIQS is based on Faircom Database. It have very different and efficient way to manage and transmit the data. And Faircom Database also helps a lot in critical section. All in all, compared with any other file system, Faircom Database shows a great advantage and it is good solution for a huge complex system.

## **Introduction**

My project is a Remote Information Query System (RIQS) that implemented by using Faircom Database. This RIQS is accessible to all clients on the same network. This network can be a local area network or an Internet. My project has three goals:

1. Establish a data table with some customers' information on the server.
2. Manage customers' records on a client via the network. The management includes search a record, update a record, delete a record or add a new record
3. Able to handle multi-requests from different clients at the same time.

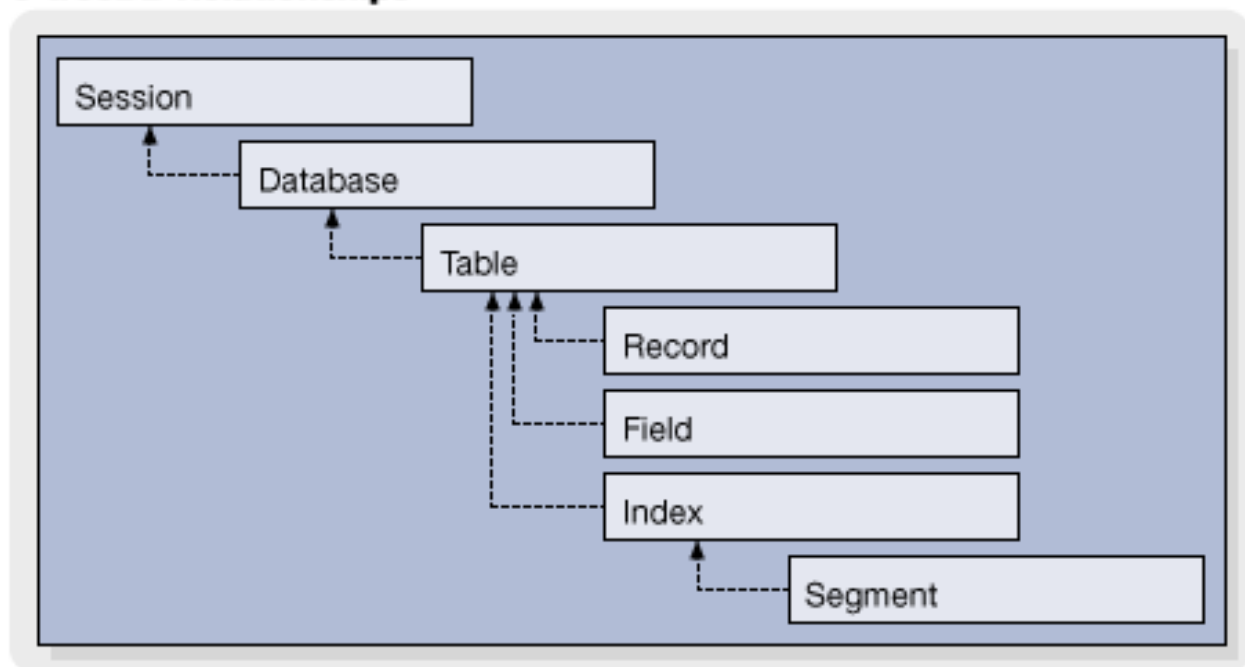
Compared with other approaches, I use Faircom database instead of file system to manage my data. By using Faircom database, we have several advantages. First of all, we don't need to design a file system by ourselves. Secondly, FairCom database provide transaction support. So we don't need to worry about critical section. Imaging that we have a huge and complex system based on file system, at a certain moment, two or more tasks/thread may try to access and update the

same record. This situation will bring a big problem. Usually we use semaphore to avoid this. But it is too difficult for a complex system like mine project. On the contrary, Faircom database can do it quite well, user just need to lock the session before the read/write operation and unlock the session after the read/write operation. The third advantage is fast indexing. The Faircom database can store the data based on some algorithm, so next time when user try to access to this data, Faircom database can find it very fast and efficiently. Last, user can create several data table on the database. And user can view the relation between each table.

## Background

c-treeACE is a cross-platform database engine developed by Faircom. Faircom currently provide the database engine for Windows, Linux, Mac OS X, PH-UX, AiX, and Solaris.

### c-treeDB Relationships



A Faircom database contains session, Database, Table, Record, Field.....

A session represents a connection between a client and a c-tree Server; any work should start with activating the session. The session handle indicates the c-treeDB session, the server name and location, the directory where the databases are located, the user name and password.

A database can be considered as a collection of tables, and each database has its own database dictionary that stores information about each table that belongs to that database

A table is essentially a c-tree Plus data, and optionally, index files. There can be, and typically are, more than one table in a database, and a given table may belong to multiple databases. A Table may have zero or more records.

A field is the basic element of a table, and a collection of fields forms a data record.

Indices typically have one or more segments describing the index key structure. The index handle indicates an index associated with a particular table, while the segment links the index with the fields.

A record is essentially an entry row in a table. A record handle indicates a record instance on a particular table. A table may have one or more record handles associated with it. Each record handle may be an independent cursor into the table, or several record handles may share the same cursor into the table.

Key API Function:

`ctdbAllocSession()`

Allocates memory and initialize a new session handle

ctdbLogon()

Logs on to the c-tree Server or c-treeACE instance.

(ctdbLogon(hSession, "FAIRCOMS@10.8.89.17", "ADMIN", "ADMIN"))

ctdbAllocRecord()

Allocates memory and initialize a new record handle.

ctdbFindRecord()

ctdbWriteRecord()

And some other useful functions:

ctdbSetFieldAsString()

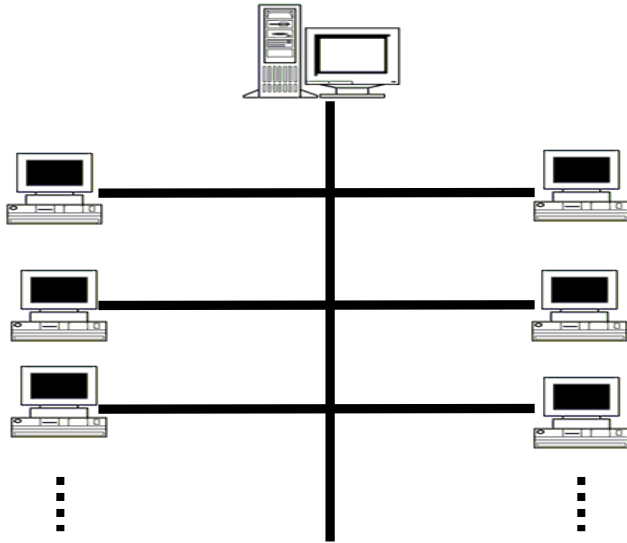
ctdbGetFieldAsString()

ctdbClearRecord()

## **Proposed Implementation**

### **Hardware:**

Several workstations with Linux working on it. All workstations have to connect to the same network. To confirm the connection, workstation can ping each other. If receive the response, then the connection is established, otherwise, there is no connection between these two workstation. The overall structure is shown as follow.



## Software

### Server

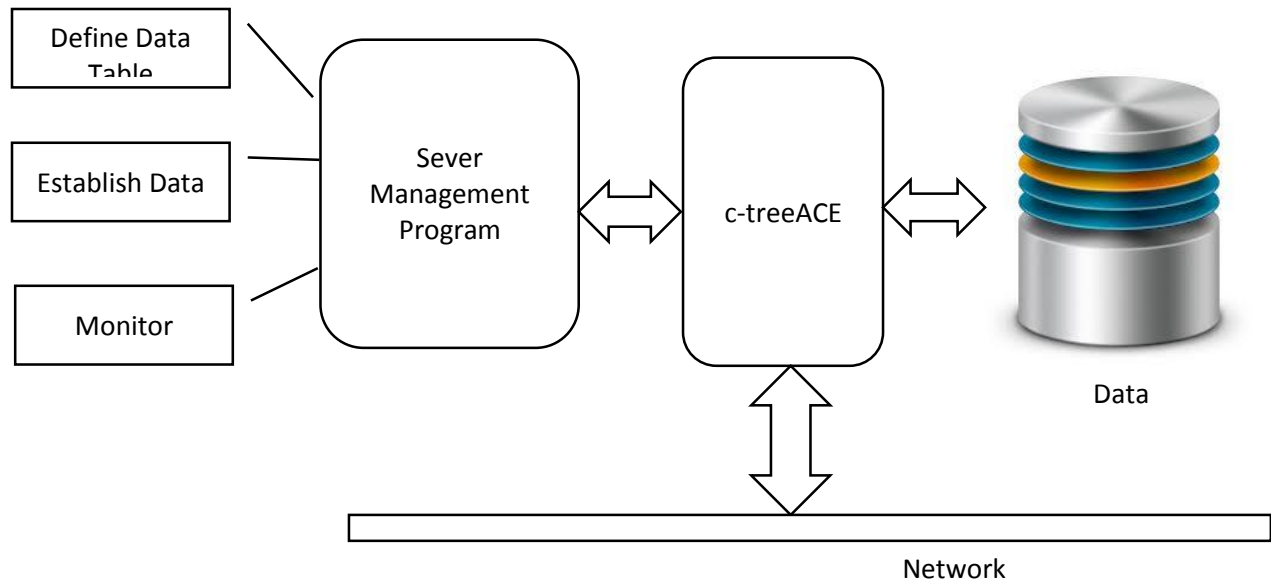
This server program is working on workstation 1. It is used to establish the first record. This program is provided to the database manager.

The first step is logon the session. Since the server program and c-treeACE is on same workstation. So we don't need to hardcode the IP address anymore. So the code is

```
ctdbLogon(hSession, "FAIRCOMS", "ADMIN", "ADMIN")
```

The next step is open the table, if they exist. Otherwise create one and open it.

The next step populate the table. By doing this, a table with customers' information is established.



## Client

Client is a program that run on workstation 2, this is the program that provided to user. A user can access and update any data from workstation 2

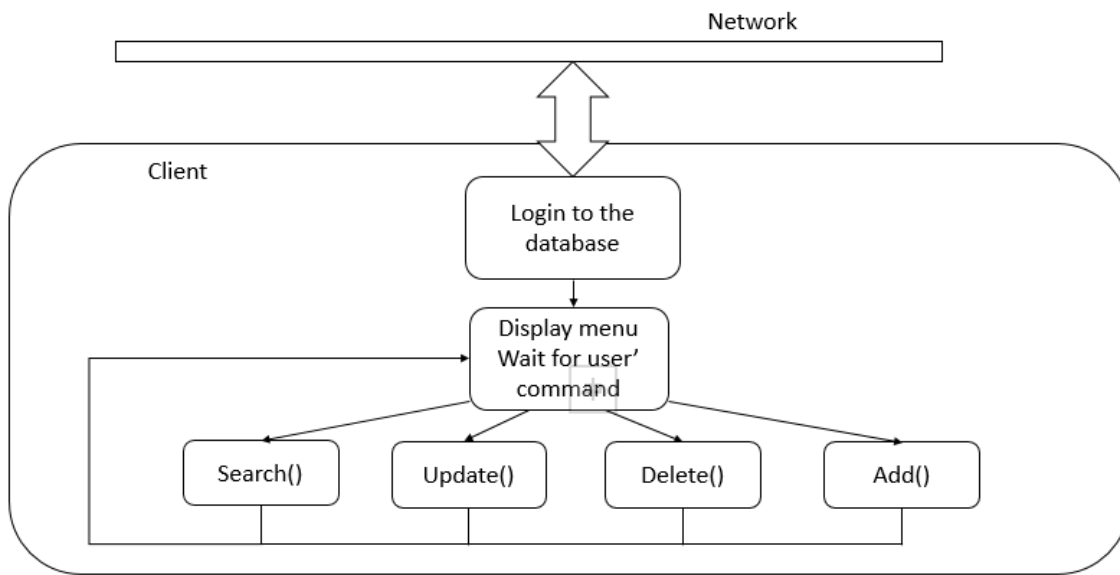
The first step is also logon the session.

```
ctdbLogon(hSession, "FAIRCOMS@XXX.XXX.XXX.XX", "ADMIN", "ADMIN")
```

where xxx.xxx.xxx.xx is the IP address of the server

The next step is open the table, if they exist. Otherwise create one and open it.

The next step is display the user's interface, and let the user to choice which operation is going to be executed next. The options includes search a record, update a record and add a new record.



## Simulate multi-request:

To simulate multi-request, execute a program on workstation 3. This program will create multiple thread and each thread will send a request and update a certain record every 0.5s.



## Results

Server:

```
INIT
Logon to server...
DEFINE
table CustomerMaster
MANAGE
Delete records...
Add records...
Display records...
1000 92867 Bryan Williams 2999 Regency,Orange 1234567890
1001 61434 Joshua Brown 13 Main,Harford 3123452423
1002 73677 Wenqlang Bo 4356 Cambridge,Atlanta 7145781667
1003 10034 Keyon Dooling 19771 Park Avenue,Columbi 3225564377
1004 42155 Jack sparrow 1600 Rolling hill,New york 4425575446
1005 87765 Kimberly Plwell 19771 Park Avenue,Columbia 6644333455
1006 33245 Randel Archibold 19771 Park Avenue,Columbia 3333333333
1007 65201 Jonathan Weisman 19771 Park Avenue,Columbia 4533464754
1008 35543 Julia Barrd 19771 Park Avenue,Columbia 8754324552
1009 44567 Henry Fountain 19771 Park Avenue,Columbia 5544766995
1020 44567 Andrea Elliott 19771 Park Avenue,Columbia 5544766995
DONE
Close table
Logout...
```

As the snapshot shows, the server initializes the database and creates a table with customers' information on it----ID, Zip code, name, address and cell phone number.

Client:

```
INIT
Logon to server...
DEFINE
table CustomerMaster
MANAGE
Myfunction
*****
Welcome to use custom database!
1.search a user by ID!
2.Updata user's information!
3.Add a record!
*****
```

Once the client program is executed, the client logons to the server first and defines an empty table, then display the user's interface and waits for the user's input.

```
*****
Welcome to use custom database!
1.search a user by ID!
2.Updata user's information!
3.Add a record!
*****
1
Search
      Update record...
input your Security ID
1002
ID      ZipCode      Name      Address      Phone
1002    73677    Wenqiang Bo    4356 Cambridge,Atlanta    7145781667
*****
```

Firstly, do searching by inputting "1" and then input the ID that you want to search, for example "1002". The program will return the record with the same ID.

```
*****
Welcome to use custom database!
1.search a user by ID!
2.Updata user's information!
3.Add a record!
*****
2
Updata
      Update record...
input your Security ID
1002
ID      ZipCode      Name      Address      Phone
1002    73677    Wenqiang Bo    4356 Cambridge,Atlanta    7145781667

Which record you want to change?
1.phone number?
2.name?
3.address?
4.delete this record
1
Please input your new phone number!
9999999999
ID      ZipCode      Name      Address      Phone
1002    73677    Wenqiang Bo    4356 Cambridge,Atlanta    9999999999
*****
```

Secondly, do updating by inputting "2" and input the ID that you want to update, for example "1002", the program will ask user to update which record. It can be

the cell phone number, name address or delete this record. Here, we update the phone number to “9999999999” and later update the name to “abc”.

```
*****
Welcome to use custom database!
1.search a user by ID!
2.Updata user's information!
3.Add a record!
*****
1
Search
    Update record...
input your Security ID
1002
ID      ZipCode   Name                Address              Phone
1002    73677     abc                 4356 Cambridge,Atlanta  9999999999
```

After the updating, search such record again, and you can see the record is changed.

```
*****
2
Updata
    Update record...
input your Security ID
1002
ID      ZipCode   Name                Address              Phone
1002    73677     abc                 4356 Cambridge,Atlanta  9999999999

Which record you want to change?
1.phone number?
2.name?
3.address?
4.delete this record
4
Do you really want to delete this record?y
ID      ZipCode   Name                Address              Phone
1002    73677     abc                 4356 Cambridge,Atlanta  9999999999
*****
Welcome to use custom database!
1.search a user by ID!
2.Updata user's information!
3.Add a record!
*****
1
Search
    Update record...
input your Security ID
1002
Sorry, can't find this Reocrd
ID      ZipCode   Name                Address              Phone
1002
```

If we input “4”, it ask the user to confirm. By input “y”, it will delete this record. If user try to search this record. The program will return a message says “sorry, can’t find this Record”.

```
*****
Welcome to use custom database!
1.search a user by ID!
2.Update user's information!
3.Add a record!
*****
3
Add
Please input the ID!
1099
Please input the ZipCode!
11111
Please input the Name!
wang tao
Please input the Address!
jiangyin,jiangsu
Please input the Phone number!
0000000000
ID      ZipCode   Name      Address      Phone
-----
1099   11111   wang tao   jiangyin,jiangsu   0000000000
*****
Welcome to use custom database!
1.search a user by ID!
2.Update user's information!
3.Add a record!
*****
1
Search
Update record...
Input your Security ID
1099
ID      ZipCode   Name      Address      Phone
-----
1099   11111   wang tao   jiangyin,jiangsu   0000000000
*****
```

Thirdly, do adding by input “3”, the program will ask user to input a series of information, like ID, Zip code, name, address and phone number. Later, user can check the record by search this record.

## Conclusions

The project is a success and I accomplish my goals. Firstly, I successfully establish the server and create the initial data on workstation 1. Secondly, I successfully run a client program on workstation2 and display, update, add or delete the record remotely. Thirdly, by using a program running on workstation 3, I simulate the multi-query and multi-update to test the critical section. However there are many possible improvement that could be made to this implementation. First of all,

establish a more complex database with several data table. At present, my program have one data table. If I can create more tables, this database can contain more information about the customers. Next, multi-condition search. For example, if we don't remember the ID of the customer, we can search the record based on the age, sex and residence. Another improvement is code porting the client program to the embedded mobile device, like Ts7250 board. Actually I intended to write the program on Ts7250 board at the beginning. But the Faircom Database is still not working on it. Anyway, if it is accomplished, we may be able to access the database from anyway anytime.

My project has a very wide potential application. For example, distributed environment monitor that update the temperature, humidity, lightness to the server in real time. This monitor might be useful in greenhouse, indoor environment or atmospheric research. Another application is robots communication and control. Each robot don't need to communicate with each other directly. What they need to do is upload their data to the server and the server will send the command to the robots based on the data, for example robotic soccer and autonomous driving.