



Birdstep Database Insights:

Using the RDM Embedded Native Interface

Birdstep Technology's RDM Embedded database management software is a highly flexible dbms with over 150 APIs for robust database management. This paper provides a simple "How To" example using the native API set with RDM Embedded.

This article assumes that you have a solid background with developing C applications and have installed the RDM Embedded SDK.

The RDM Embedded SDK is available at http://www.birdstep.com/downloads/database_download.php3 (follow the "download free SDK" link). The API and definitions can be found in the product Reference Manual Chapter 3.

Setup

The following pieces are all that is required to access an RDM Embedded database from a C/C++ application built on the Windows platform:

- 1) MSVC 6.0 IDE.
- 2) RDM Embedded 7.x for Windows.

Once installation of these components is completed there are three simple configuration steps.

Step 1) Environment Configuration

There are two environmental settings that need to be applied in order to access your data:

- The bin/win32 directory needs to be defined in your path setting.
- The RDM Embedded library files need to be in your path. The native interface requires `rdm7[u]` and `psp7[u]`.

Step 2) Database Configuration

Access to an RDM Embedded database is governed through a C header file and an RDM dictionary file, both of which are created by the **ddl** utility that is included in the RDM Embedded SDK. For the example to follow, we will create the header file and dictionary file from an existing schemea.

The schema below describes a database called, **simpledb**, with one table called **simpletb**. The **simpletb** table consists of 3 fields: the 1st is a key field called **fld1** with *long* elements, the 2nd is **fld2** with *short* elements and the 3rd is **fld3** which is an array of *char[10]*. Finally **datafile1.D01** will contain the contents of the database and **keyfile1.K01** will contain the indexed elements of field **fld1**.



To create the header and dictionary files, first copy the schema shown below into a text file called `schema.ddl`. Next, within a DOS box, run the `ddlp` utility against the schema file: ***ddlp schema.ddl***.

```
database simpledb
{
    data file one = "datafile1.D01" contains simpletb;
    key   file two = "keyfile1.K01" contains fld1;

    record simpletb
    {
        key long fld1;
        short fld2;
        char fld3[10];
    }
}
```

Through the `ddlp` utility you will have created two new files:

- C header file: **simpledb.h**
- RDM Embedded dictionary file: **simpledb.dbd**

Step 3) Application

To prepare an application to use the native interface, include the following *include* statements in your source files:

```
#include <stdio.h>
#include "rdm.h"
#include "simpledb.h"
```

Your application is now configured to access your RDM Embedded data.

My First RDM Application

We will now create a very simple "Hello World" application. The application will consist of the following sections:

1. Defining the lock manager.
2. Initializing the database.
3. Populating the database.
4. Scanning the database via two navigation methods:
 - a. Scanning through individual record instances
 - b. Scanning via the indexed field
5. Closing the database
6. Compiling
7. Lock manager
8. Execution

Section 1: Defining the lockmanager

For multi-user access to an RDM Embedded database, you will need to define your lock manager interface. There are two lock managers available; the internal lock manager for environments in which all the users are located on the same machine, and the TCP lock manager where clients are located across a network. For the example the latter, using the `d_lockcomm` API, is selected:

```
d_lockcomm(psp_lmFind("TCP"), DEFAULT_TASK)
```

Next select the named lock manager the with which the application will be communicating. This is done using the `d_lockmgr` API, where the 2nd parameter is the IP address of the machine where the lock manager process is running. More details will be given on the separate lockmanager process later on.

```
d_lockmgr("192.168.2.97", DEFAULT_TASK) <note: your IP address should be used>
```

Finally, define a user name using the `d_dbuserid` API.

```
d_dbuserid("TheUser", DEFAULT_TASK)
```

Section 2: Initializing the database

In order to initialize the database, first open the database in exclusive mode using the `d_open` API and then initialize the database using the `d_initialize` API. Finally close the database, so that it can be opened later in a shared mode.

```
d_open("simpledb", "x", DEFAULT_TASK)
```

```
d_initialize(DEFAULT_TASK, CURR_DB)
```

```
d_close(DEFAULT_TASK)
```

Section 3: Populating the database

Now that the db is initialized, the table, **simpletb** can be populated. The proper steps are: start a transaction, request a lock, add the new data, and commit the transaction.

Start a new transaction, called "Enter":

```
d_trbegin("Enter", DEFAULT_TASK)
```

Place a write lock on the table, **simpletb**:

```
d_relock(SIMPLETB, "w", DEFAULT_TASK, CURR_DB)
```

Fill the individual elements:

```
while (j <= 100)
{
```

```

        i_simpletb.fld1 = rand();
        i_simpletb.fld2 = j;
        strcpy(i_simpletb.fld3, "Hello World!");

        d_fillnew(SIMPLETB, &i_simpletb, DEFAULT_TASK, CURR_DB)
        j++;
    }

```

End the transaction, resulting in all the locks being released, and any changes being committed to database.

```
d_trend(DEFAULT_TASK)
```

Section 4: Navigating through the database

Section 4a: Navigating via record instances

To demonstrate the Navigation procedures, a database record navigation will be performed to confirm the data was properly entered in Section 3. Working within a transaction (this time called "Record"), a "read" lock on the table will be used. To simplify the scanning process, use the `d_recfirst` and `d_recnext` API's inside a loop and print out the contents of field one and field two:

```

    // Starting a new transaction: "Record"
    d_trbegin("Record", DEFAULT_TASK)

    // Placing "read" lock on SIMPLETB
    d_recllock(SIMPLETB, "r", DEFAULT_TASK, CURR_DB)

    for (status = d_recfirst(SIMPLETB, DEFAULT_TASK, CURR_DB);
        status == S_OKAY;
        status = d_recnext(DEFAULT_TASK, CURR_DB))
    {
        d_recread(&j_simpletb, DEFAULT_TASK, CURR_DB)
        printf("\tfield 1: %d \tfield 2: %d\n", i_simpletb.fld1,
            i_simpletb.fld2);
    }

    /* End transaction and release all locks */
    d_trend(DEFAULT_TASK)

```

One thing to note is that this method of scanning illustrates the network model architecture of RDM Embedded, which in its simplest terms is just a linked list. Each of the records printed are in the same order as the linked list.

Section 4b: Navigating via an indexed field

Another navigation example is to navigate the database through the indexed field fld1. Again working within a transaction (this time called “Key”), a “read” lock on the table will be used. To simplify the scanning process, use the d_keyfirst and d_keynext API’s inside a loop and print out the contents of field one and field two:

```
// Starting a new transaction: "Key"
d_trbegin("Key", DEFAULT_TASK)

// Placing "read" lock on SIMPLETB
d_relock(SIMPLETB, "r", DEFAULT_TASK, CURR_DB)

for (status = d_keyfirst(FLD1, DEFAULT_TASK, CURR_DB); status ==
S_OKAY;
status = d_keynext(FLD1, DEFAULT_TASK, CURR_DB))
{
    d_recread(&i_simpletb, DEFAULT_TASK, CURR_DB)
    printf("\tfield 1: %d \tfield 2: %d\n", i_simpletb.fld1,
i_simpletb.fld2);
}

/* End transaction and release all locks */
d_trend(DEFAULT_TASK)
```

Section 5: Closing the database

The last thing to do is close the database using the d_close API:

```
d_close(DEFAULT_TASK)
```

Section 6: Compilation

Having completed the source code for my application, the database can be compiled. This example uses the MSVC 6.0 IDE – simply press <F7> to compile. Below are some common errors that from the compilation process:

Possible Compilation Errors

- | | |
|-----------|---|
| Error: | Cannot open include file: 'rdm.h' |
| Solution: | Make sure you have defined in your project settings the correct include path for the rdm.h header file. If you did a standard installation, it should be located in C:\Program Files\Birdstep Technology\RDM Embedded 7.0\include |
| Error: | Unresolved external symbol _psp_lmcFindA@4 |
| Solution: | You did not include the psp7[u] library in your path settings. This library is required if you wish to have multi-user access to an RDM Embedded database. |



Section 7: Lock Manager

Before executing the HelloWorld application, start up the lock manager. This is a separate process that runs on the same machine as the application, though this is not necessary for the TCP lockmanager. To initiate, simply type "lm" from within a DOS box. The following message will be returned:

```
Lock Manager
RDM Embedded 7.0.367 [16-Jul-2003] http://www.birdstep.com
Copyright (c) 1992-2003 Birdstep Technology, Inc. All Rights
Reserved.
```

```
TCP/IP sockets Lock manager
Lock Manager - '192.168.2.97'
```

The IP address is the address used with the d_lockmgr API in Section 1. Your IP address will be different.

Section 8: Execution

Finally all that is left is execution. From the MSVC IDE, simply type <ctrl><F5>. A few common errors that may be experienced are:

If everything is correctly done, your output should like the following

```
SCANNING DATABASE BY RECORD INSTANCES
    field 1: 41      field 2: 1
    field 1: 18467  field 2: 2
    field 1: 6334   field 2: 3
    field 1: 26500  field 2: 4
    field 1: 19169  field 2: 5
total of 5 simpletb records in database

SCANNING DATABASE BY KEY FIELD: fld1
    field 1: 41      field 2: 1
    field 1: 6334   field 2: 3
    field 1: 18467  field 2: 2
    field 1: 19169  field 2: 5
    field 1: 26500  field 2: 4
total of 5 simpletb records in database
```



Possible Runtime Errors

- Error: TCP/IP connect fail: (10061), SYSTEM/OS error: -920, no lock manager is installed
- Solution (1): Make sure you have started your lockmanager process.
- Solution (2): Make sure the IP address in the d_lockmgr API is the same as IP address where the lock manager process is running.
- Error: SYSTEM/OS error: -944, TAF-lockmgr synchronization
- Solution: Delete the rdm.taf file.

Complete Source

The source for the MSVC 6.0 project files can be found [here](#).

Did you find this article helpful? Please let us know at americas@birdstep.com