

СОЗДАНИЕ API ДЛЯ УНИФИЦИРОВАННОГО ВЗАИМОДЕЙСТВИЯ ПО «МОНИТОР РЕАЛЬНОГО ВРЕМЕНИ» С БАЗОЙ ДАННЫХ *SQLITE*

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

В данной статье рассматривается разработка и реализация унифицированного механизма взаимодействия ПО «Монитор реального времени» с БД *SQLite*.

«Монитор Реального Времени» — управляющая программа центральной приемо-передающей станции (ЦППС) «СИСТЕЛ» и МТК-30.КП (устройство телемеханики контролируемого пункта), предназначенная для обеспечения функционирования систем сбора и первичной обработки данных в составе Автоматизированных систем диспетчерского управления (АСДУ).

ПО «Монитор» обеспечивает:

- прием и передачу данных (телеинформации, команд телеуправления и телерегулирования, данных от счетчиков электрической энергии) по широкому спектру протоколов;
- первичную обработку данных;
- формирование массивов для передачи на верхний уровень в соответствии с заданными протоколами обмена;
- синхронизацию времени подключенных устройств телемеханики.

Программа «Монитор» работает под управлением операционных систем *MS Windows 2000/XP/Server 2003*, а также под управлением ОС *Linux Fedora Core 6/ Linux Debian 4* и выше.

Функционально-программный комплекс «Монитор» можно разделить на следующие составляющие (рис. 1):

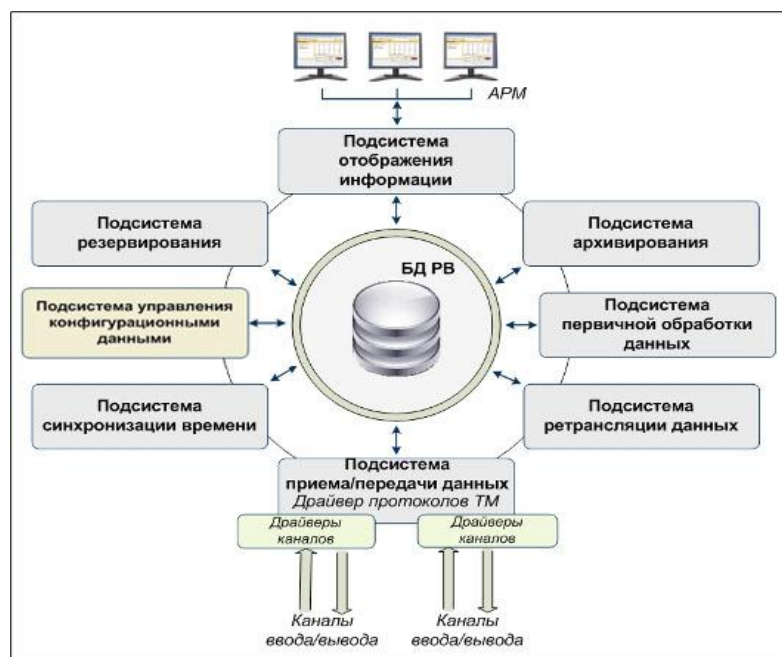


Рис. 1. Структура программного комплекса «Монитор»

Программное обеспечение «Монитор реального времени» используется для сбора данных с объектов телемеханизации и управления этими объектами. Конфигурация указанного программного обеспечения включает описание устройств сопряжения, каналов связи, принимаемых и рассчитываемых телемеханических сигналов. Параметризация программного обеспечения «Монитор реального времени» осуществляется посредством текстовых файлов и *DBF*-таблиц.

На данный момент рассматриваемое программное обеспечение поддерживает большое количество конфигурационных файлов, что значительно усложняет его настройку и последующую работу.

Целью работы является создание механизма унифицированного взаимодействия ПО «Монитор реального времени» с БД *SQLite*.

SQLite — легковесная встраиваемая реляционная база данных. *SQLite* предоставляет библиотеку, с которой программа компонуется, и база данных становится составной частью программы. В качестве протокола обмена используются вызовы функций (*API*) библиотеки *SQLite*.

Такой подход уменьшает накладные расходы, время отклика и упрощает программу. *SQLite* хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа.

Основные преимущества использования *SQLite*:

- простота использования. При создании таблицы необходимо только указать имена полей, но не их тип, так как все данные *SQLite* хранит в формате строки;
- работа с данными БД и интерфейс к нему реализованы как единое целое в виде одной библиотеки. Огромным преимуществом *SQLite* является высокая производительность;
- легкость администрирования. *SQLite* хранит данные в обычных файлах, и отпадает всякая необходимость в дополнительных средствах администрирования;
- легкая переносимость между платформами, веб-серверами и приложениями. Файлы баз данных совместимы с различными платформами (*Windows, UNIX*).

Данная работа связана с организацией перехода ПО «Монитор РВ» от работы с *DBF*-файлами к работе с базами данных *SQLite* без внесения значительных изменений.

Существуют несколько классов, которые работают с *DBF*-файлами: *MwDbf, cdbfile*. В этих классах описаны функции:

- открытия файла *BOOL Open(const MwString& TableName);*
- закрытия файла *void Close(void);*
- подсчета количества записей и столбцов *int GetRecordCount(void ; int GetFieldCount(void);*
- взятия первой строки и следующей *void MoveFirst(void); void MoveNext(void);*
- определения имени таблицы *CString GetName(void);*
- взятия текущей строки *int GetCurrentRecord(void);*
- определения типа столбца *char GetFieldType(unsigned short FieldNum);*
- определения размера столбца *unsigned char GetFieldLen(unsigned short FieldNum);*
- определения имени столбца *CString GetFieldName(unsigned short FieldNum);*
- получения значения столбца по имени/номеру столбца *CString GetFieldBuf(const CString& Field), CString GetFieldBuf(unsigned short FieldNum);*
- перехода к строке по номеру строки *void MoveToRecord(int RecordNum);*
- добавления новой строки *void CreateNewRecord(void);*
- удаления заданной строки *void RemoveCurrentRecord(int RecordNum);*

Главная задача заключалась в разработке класса *MwSqliteDbT* с аналогичными функциями и параметрами.

Пример функции открытия *DBF*-файла:

```
BOOL MwDbf::Open( const MwString& TableName)
{
    Name = TableName;
    // printf("Open DBF - %s\n", (const char*) TableName);
    return Base.OpenFile( (const char*) TableName);
}
```

Соответствующая функция класса *MwSqliteDbT* выглядит так:

```
BOOL MwSqliteDbT::Open(const CString& TableName)
{
    Name = TableName;
    Name.MakeLower();
    Name.Replace(".dbf", "");

    int Length = strlen(TableName);
```

```

char* buf = new char[Length+15];
sqlite3_exec(db, "BEGIN;", NULL, NULL, &errmsg);
sprintf(buf, "select * from %s", (const char*)(LPCTSTR)TableName);
sql = buf;
if (sqlite3_prepare(db, sql, -1, &pStmt, NULL))
{
    PrintError();
    if (pStmt != NULL)
        sqlite3_finalize(pStmt);
    pStmt = NULL;
    return FALSE;
}
CurrentRecord = 0;
FieldCount = 0;
FieldCount = sqlite3_column_count(pStmt);
GetRecordCount();
delete buf;
return TRUE;}

```

Внутри функции переменной *Name* присваивается имя таблицы, например *Calibr.dbf*, а потом выполняется преобразование этого имени. Все буквы становятся строчными и расширение *.dbf* удаляется. Имя становится *calibr*. Чтобы открыть таблицу необходимо выполнить запрос «*select * from TableName*». Прежде чем выполнить *SQL* запрос, его необходимо подготовить к выполнению, откомпилировав в байт-код с помощью функции *int sqlite3_prepare()*. После компиляции *SQL* запроса ее можно выполнить с помощью одного или нескольких вызовов функции *int sqlite3_step (sqlite3_stmt *pStmt)*. Таким образом, оставляем названия и параметры функций неизменными, а заменяем только тела функций.

Результатом проделанной работы является создание специального программного обеспечения для настройки и работы ПО «Монитор РВ» с данными, хранящимися в базе данных *SQLite*.

Библиографический список

1. Документация «Монитор реального времени системы сбора и первичной обработки телеинформации. Руководство системного программиста», 149 с.
2. <http://www.sqlite.org/docs.html> (Руководство по *SQLite*).