ГОСУДАРСТВЕННЫЙ НАУЧНЫЙ ЦЕНТР РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

ИФВЭ 2011–26
ОЭФ

И.В. Лобов, А.В. Лутчев

Управление технологическим оборудованием с использованием реляционной базы данных

Протвино 2011

УДК 004.424 М-24

Аннотация

Лобов И.В., Лутчев А.В. Управление технологическим оборудованием с использованием реляционной базы данных: Препринт ИФВЭ 2011–26. – Протвино, 2011. – 16 с., 7 рис., 2 табл., библиогр.: 2.

Предлагается метод решения задачи управления оборудованием в условиях, когда требуется организация удаленного управления с разных точек вычислительной сети, а также когда требуется разделение прав на управление. Метод основан на использовании баз данных. Техническим результатом предлагаемого метода является высокая скорость разработки программного обеспечения и простота организации управления, поскольку используются штатные средства систем управления реляционными базами данных.

Abstract

Lobov I.V., Lutchev A.V. The Database Driven Technological Equipment Control: IHEP Preprint 2011–26. – Protvino, 2011. – p. 16, figs. 2, tables. 2, refs.: 2.

The software method for U-70 technological equipment control implementation is proposed. The method is intended for the case when multiply asynchronous remote control with computer authentication is assumed. The main underlying principle of the control algorithm is based on the utilization of the standard database features. The technical results of the proposed approach is both high rate of the software development and simplicity of the control process because of standard database facilities being used.

Введение

Под *технологическим оборудованием* (далее просто «оборудование») в настоящей работе понимаются любые механические или электрические устройства, способные изменять свое состояние под управлением аналоговых воздействий или цифровых сигналов.

Под *управлением* далее будет пониматься компьютерная программа (далее «управляющая программа»), способная изменять состояние оборудования. Термин «программа» является собирательным, в простейшем случае это действительно одна программа, а в реальности это может быть *управляющая система* - набор программ, взаимодействующих между собой сложным образом. Существует множество способов построения таких управляющих систем. В настоящей работе предлагается способ, базирующийся на использовании реляционных баз данных, системы управления (СУБД) которых имеют следующие особенности:

- возможность удаленного сетевого доступа;
- наличие хранимых процедур;
- транзакционная форма выполнения запросов.

1. Схема управляющей системы

Рассмотрим простой пример управления источником питания (ИП). Управляющая программа должна передать в источник питания уставку напряжения, а источник питания в свою очередь должен выдать напряжения, соответствующее этой уставке. Записав уставку в источник питания, оператор должен увидеть результат выполнения своей команды – выдал ли источник требуемое напряжение или нет. Поэтому операцию записи уставки в ИП необходимо дополнить обратной связью – производить периодическое чтение данных с ИП. Внешний вид программы приведен на Рис.1. Графический индикатор показывает величину напряжения на выходе ИП, а при нажатии кнопки «Выполнить» происходит передача в ИП заданной уставки напряжения.

Рис 1. Внешний вид окна программы.

Алгоритм программы приведен на Рис. 2:

```
//---------- Управляющая программа ------
extern int flgButton, float uset;   // данные граф.подсистемы
void main()
{
      float data;

      while(1) {
            if (flgButton) {          // проверка нажатия на кнопку
                  flgButton = 0;      // сброс флага нажатия на кнопку
                  putCommand (uset); // посылка уставки напряжения в ИП
                  }

            data = getData();  // чтение напряжения из ИП
```

```
            drawData(data);     // отображение на экране полученных данных
            }
}
//---------------------------
void putCommand (float uset)
{
      waitReadyPut()          // ожидание готовности передачи в ИП
      put (uset);             // передача команды в ИП (uset)
}
//---------------------------
float getData ()
{
      float u;
      waitReadyGet();         // ожидание прибытия данных
      u = get();              // чтение данных от ИП
      return u;
}
```

Рис. 2. Программа управления одним ИП.

Программа в бесконечном цикле производит чтение текущего напряжения из ИП. При нажатии кнопки записи уставки напряжения взводится флаг `flgButton` и производится запись уставки в ИП. Переменные `flgButton` и `uset` предоставляются исполняемой системой графического интерфейса: флаг `flgButton` устанавливается при нажатии оператором на кнопку «Выполнить», а переменная `uset` содержит уставку напряжения, введенную оператором в поле ввода.

Процедуры `putCommand` и `getData` взаимодействуют с оборудованием путем опроса аппаратных флагов готовности (это обычная схема обмена данными программы с оборудованием) и, вследствие этого, могут занимать некоторое время на свое выполнение – от нескольких микросекунд до сотен миллисекунд. Наиболее длительной по времени является процедура `getData` из-за ожидания прибытия данных `waitReadyGet`. На Рис. 2 приведены упрощенные тексты этих процедур, без обработки ситуаций тайм-аута, возникающих при сбоях передачи данных по линии связи.

На Рис. 2 приведен способ т.н. синхронного выполнения запросов к оборудованию, когда операции чтения/записи данных в оборудование производятся

программой последовательно и, тем самым, тормозят ее выполнение. Внешне это проявляется в том, что программа периодически «зависает» на время выполнения процедуры `getData`, а после нажатия кнопки «Выполнить» происходит ожидание завершения выполнения процедуры `putCommand`.

Программа непрерывно (цикл `while`) считывает данные с ИП, поскольку оператор должен все время иметь перед глазами текущее значение напряжения, выдаваемого ИП. Можно, конечно, оптимизировать код приведенной выше программы, сделав чтение данных не непрерывным, а периодическим с периодом, скажем, в 9 секунд. Однако в любом случае программа будет «замирать» на время выполнения процедуры `getData`. Это весьма серьезное неудобство синхронной схемы управления, оно особенно проявляется в случае, когда источников питания несколько. В этом случае оператор не сможет передать уставку в ИП, пока не завершатся операции чтения данных всех ИП (см. рис. 3).

```
//---------- Управляющая программа ------
extern int flgButton[], float uset[];
#define NPWS      100;              // кол-во ИП
void main()
{
      int i;
      float data;

      while(1) {
            for (i=0; i < NPWS; i++)
                  if (flgButton[i]) {
                        flgButton[i] = 0;
                        putCommand (uset[i]);   // задержка!
                        }

            for (i=0; i < NPWS; i++) {
                  data = getData();             // большая задержка!
                  drawData(i, data);
                  }
            }
}
```

Рис. 3. Программа управления несколькими ИП.

Значительно более гибким является асинхронный способ выполнения запросов, когда управляющая программа разделена на две части – пользовательский интерфейс и драйвер. Пользовательский интерфейс содержит все основные операции по обработке и визуализации информации, причем выполнение кода производится без всяких задержек. А драйвер выполняет в фоновом режиме запросы от программы к оборудованию. Приведенный на Рис. 2 алгоритм управления одним ИП принимает вид:

```
//---------- Пользовательский интерфейс ------
extern int flgButton, float uset;   // данные граф.подсистемы
extern int flgPut, flgGet;          // обмен данными с драйвером
extern float dataPut, dataGet;      // обмен данными с драйвером
void main()
{
      float data;

      while(1) {
            if (flgButton) {        // проверка нажатия на кнопку
                  flgButton = 0;    // сброс флага нажатия на кнопку
                  dataPut = uset;
                  flgPut = 1;       // передаем команду драйверу
                  }

            if (flgGet){            // проверка данных от драйвера
                  flgGet = 0;       // сброс флага
                  data = dataGet;   // чтение напряжения из драйвера
                  drawData(data);   // отображ. на экране полученных данных
                  }
            }
}

//---------- Драйвер -------------------------
extern int flgPut, flgGet;          // обмен данными с польз.интерфейсом
extern float dataPut, dataGet;      // обмен данными с польз.интерфейсом
void procedure driver()
{
```

```
      while(1) {
            putCommand ();     // посылка уставки напряжения в ИП
            readData();        // чтение напряжения из ИП
            }
}
//---------------------------
void putCommand ()
{
      if (!flgPut) return;          // проверка наличия команды на запись
      if (!testReadyPut()) return;  // проверка аппаратной готовности
                                    // записи уставки напряжения в ИП
      flgPut = 0;                   // сброс команды
      put (dataPut);                // запись команды в ИП
}
//---------------------------
void getData ()
{
      if (!testReadyGet()) return 0;  // ожидание данных от драйвера
      dataGet = get();                // чтение данных от ИП
      flgGet = 1;
}
```

Рис. 4. Пользовательский интерфейс и драйвер управления оборудованием.

Драйвер по отношению к пользовательскому интерфейсу должен функционировать как параллельный процесс-нить (thread), но вызовы соответствующих процедур организации нити не приведены в тексте программы, т.к. зависят от типа операционной системы. Обмен данными между пользовательской частью и драйвером осуществляется через флаги `flgGet`, `flgPut` и данные `dataGet`, `dataPut`. Конечно, эти четыре переменные должны быть организованы с помощью мьютексов как неделимые (атомные), поскольку доступ к ним производится асинхронно из двух параллельных процессов (нитей).

Удобство схемы деления программы на пользовательский интерфейс и драйвер состоит еще и в том, что при изменении каких-либо аппаратных компонентов связи между компьютером и оборудованием придется изменить только драйвер, в то время как пользовательский интерфейс остается прежним.

Параллельность функционирования пользовательского интерфейса и драйвера имеет своим результатом то, что пользовательский интерфейс не «зависает» на время выполнения операции обмена данными с оборудованием. Вследствие этого оператор может выполнять какие-либо другие действия по управлению оборудованием в то время, когда драйвер передает запросы оборудованию и принимает от него ответы.

На Рис. 5 приведена схема управляющей системы с использованием драйвера. Направления стрелок показывают направления команд.

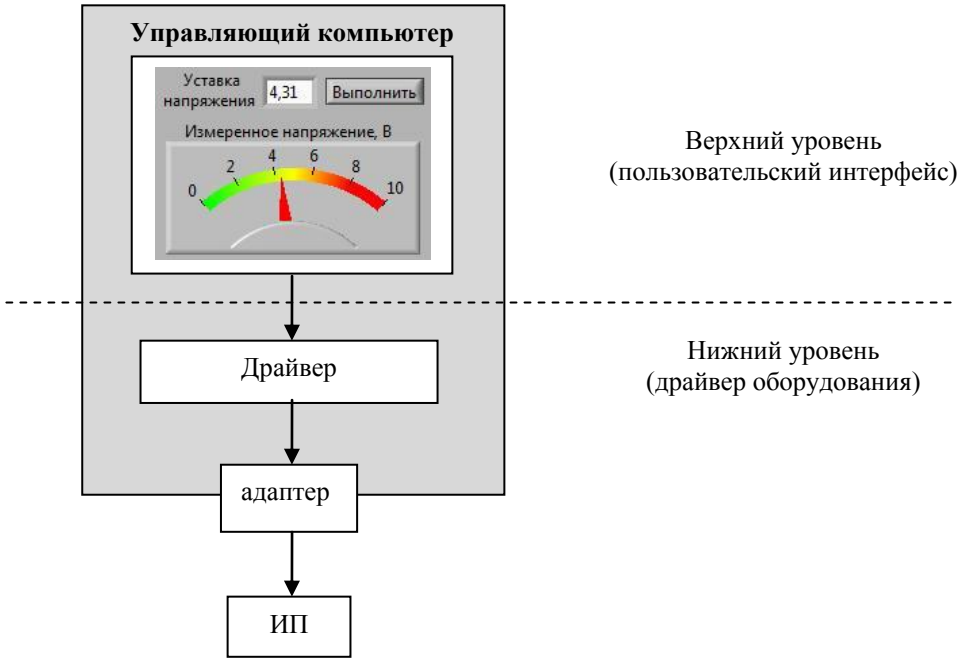

Рис. 5. Схема управляющей системы.

Взаимодействие между пользовательским интерфейсом и драйвером сводится к передаче команды сверху вниз и данных снизу вверх. Инструменты для построения обоих компонентов часто совершенно разные. Для построения драйвера в большинстве случаев требуется C/C++ и никакого графического интерфейса не требуется. А вот для построения пользовательского интерфейса нужен инструментарий типа LabVIEW, Visual Basic и т.п., позволяющий создавать удобные и красивые программы. Часто эти два компонента управляющей системы разрабатываются разными программистами.

Ниже рассматривается эволюция этой простейшей схемы управляющей системы при усложнении условий ее использования в применении к реальным ситуациям.

2. Диспетчеризация нескольких удаленных пользователей

Усложним схему Рис. 5 и предположим следующее:

- к управляющему компьютеру подключено несколько управляемых единиц оборудования (источников питания);
- имеется несколько удаленных пользователей, желающих управлять этим оборудованием, причем каждый пользователь имеет право управлять только «своим» источником питания.

В этом случае оба компонента управляющей программы разнесены пространственно: программа нижнего уровня (драйвер) работает на управляющем компьютере, а программа верхнего уровня (пользовательский интерфейс) работает на компьютерах пользователей. Управляющий компьютер превращается теперь в компьютер переднего края – КПК (в англоязычной аббревиатуре FEC – front-end computer) и становится сервером, а компьютеры пользователей становятся клиентами КПК (Рис. 6).

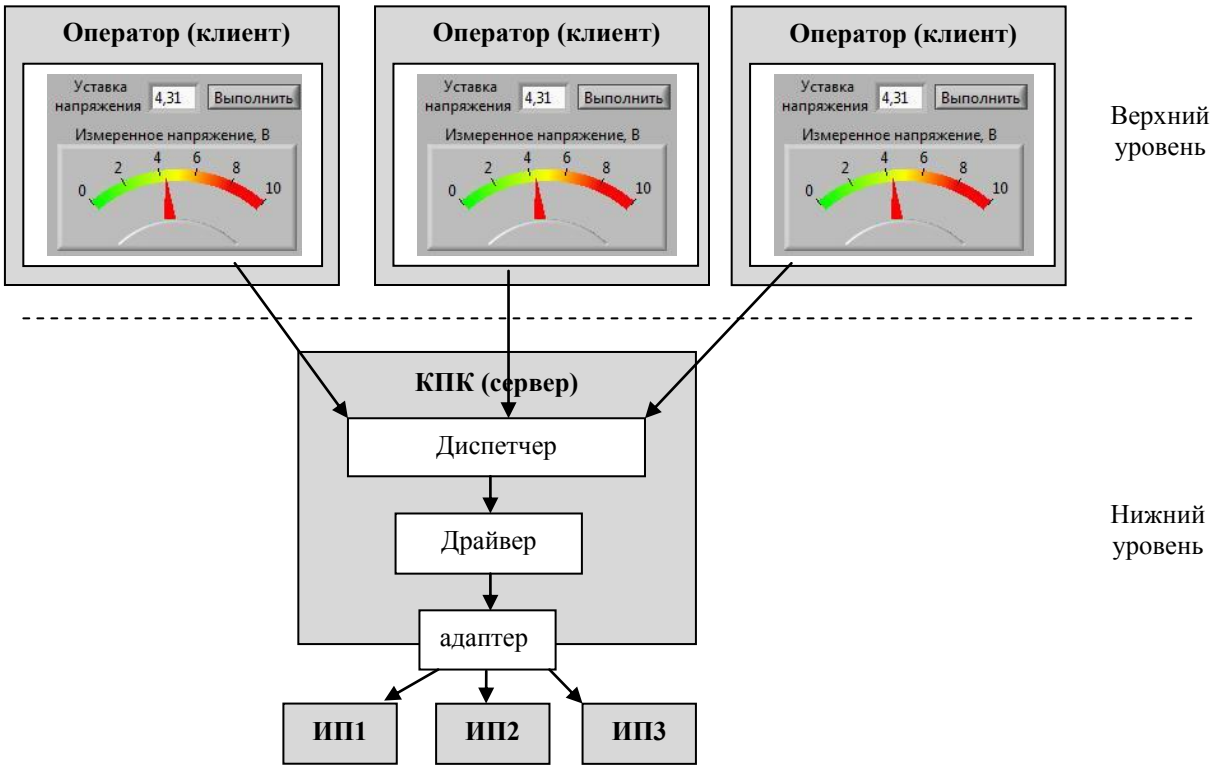

Рис. 6. Схема управляющей системы в случае нескольких удаленных клиентов с диспетчеризацией запросов.

Функции КПК заключаются в следующем:

1. Осуществлять прием запросов удаленных клиентов, организовывать принятые запросы в очередь к драйверу и отправлять ответные данные обратно клиентам.
2. Назначать клиентам различные права на управление оборудованием. Поскольку каждый клиент должен управлять только своей частью оборудования, то КПК должен разрешать пользователю управлять «своим» оборудованием и не разрешать управлять «чужим». В реальности же требуется введение более сложных прав на управление:
    - все пользователи могут просматривать состояние любого оборудования;
    - оператор местного пульта может управлять только «своим» оборудованием;
    - оператор главного пульта может управлять любым оборудованием.
3. Предоставлять клиентам возможность просмотра текущего состояния оборудования.
4. Вести архив данных и предоставлять клиентам возможность выборки и просмотра архивных данных.
5. Выполнять собственно управление оборудованием – передавать в оборудование, уставки и команды и считывать с оборудования данные.
6. Вести непрерывный фоновый опрос оборудования (полинг) с целью отслеживания текущего состояния и аварийных ситуаций.

Для выполнения перечисленных задач на КПК, помимо драйвера оборудования, должна функционировать программа диспетчеризации (диспетчер). С одной стороны, диспетчер должен уметь взаимодействовать с удаленными компьютерами по локальной сети. С другой стороны, диспетчер должен передавать команды и уставки от пользовательских запросов к драйверу и передавать назад пользователям данные оборудования, полученные от драйвера. Написание такой программы-диспетчера требует высокой квалификации программиста.

3. Использование БД

Особенностью приведенной на Рис. 6 схемы является то, что программа-диспетчер и драйвер работают параллельно на одном компьютере и использует одни и те же ресурсы (оперативную память и процессорное время). Это означает, что если программа-диспетчер начнет потреблять значительные ресурсы, то это повлияет на качество работы драйвера. Однако драйвер является более ответственной частью программного обеспечения КПК, чем диспетчер. Если диспетчер на какое-то время прекратит свою работу, это приведет только к невозможности управления оборудованием со стороны удаленных операторов. Однако останется возможным управления непосредственно с пульта КПК. А вот прекращение работы драйвера приведет к прекращению обмена данными с оборудованием, из-за чего прекратится поступление информация о его текущем состоянии.

Таким образом, важно, чтобы диспетчер не мешал работе драйвера. Однако в реальности с ростом числа клиентских компьютеров неизбежно будет расти нагрузка на программу-диспетчер, что, в свою очередь, повлияет на качество работы драйвера. Получается, что подключаясь к управляющему компьютеру и желая просто наблюдать за состоянием оборудования, клиенты фактически будут вмешиваться в процесс управления оборудованием.

По сути дела, диспетчер не имеет отношения к обмену данными с оборудованием. Его задачей является манипуляция данными и доступом к ним: хранение данных в некоем промежуточном кэше, обеспечение сетевого обмена данными, обеспечение прав доступа к данным. Именно эти задачи решают системы управления базами данных (СУБД). Возникает идея поручить КПК выполнять только операции управления оборудованием, а все операции взаимодействия с удаленными клиентами поручить выделенному серверу с базой данных. Схема управления принимает следующий вид (рис. 7).

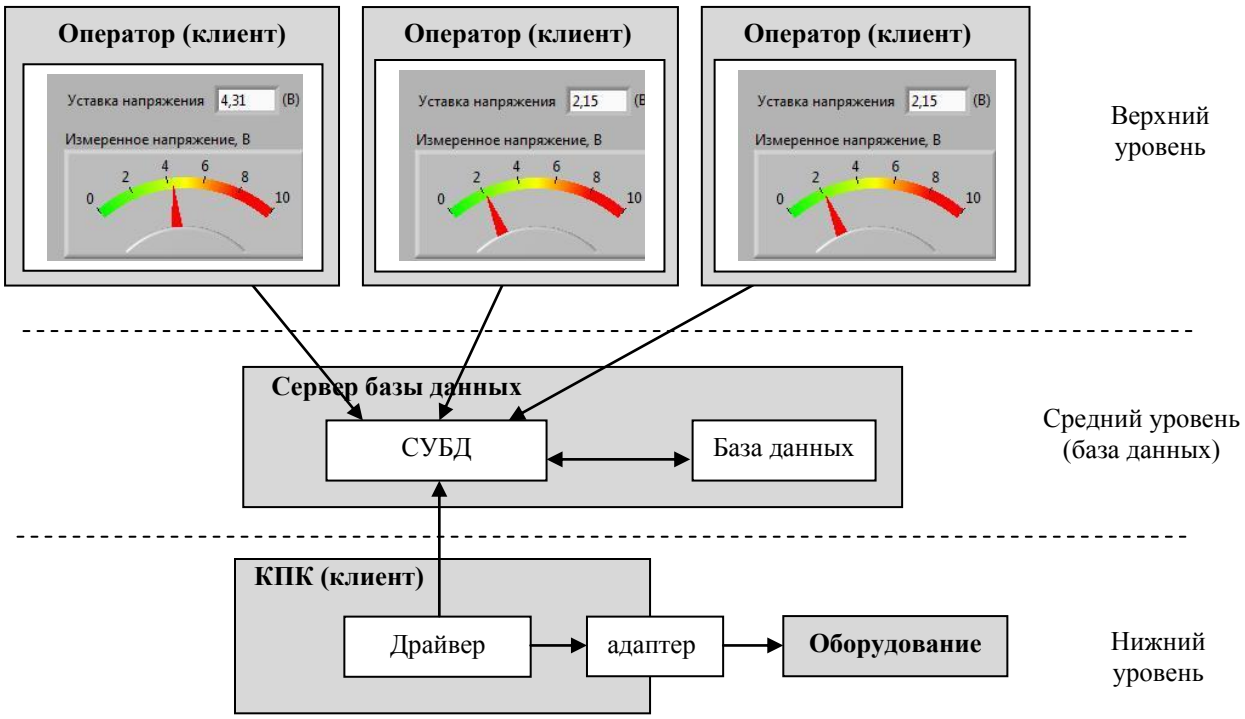

Рис. 7. Схема управляющей системы в случае нескольких удаленных клиентов с использованием СУБД.

СУБД заменяет собой программу-диспетчера и располагается на выделенном сервере в качестве промежуточного звена между множеством клиентов и компьютером переднего края. Удаленные операторы и КПК являются клиентами сервера БД.

Таким образом, из перечисленного в п.2 набора функций КПК, операции 1-4 выполняет сервер базы данных, а КПК выполняет только операции 5 и 6, причем делает это в удобное для себя время. Тем самым, рост количества удаленных клиентов не оказывает никакого влияния на работу КПК.

4. Описание логики управления с использованием БД

На сервер БД возложена задача поддержки *виртуальной копии* состояния технологического оборудования. Иными словами, сервер должен содержать постоянно обновляющиеся данные, отражающие текущее состояние оборудования. Период обновления данных определяется характером работы оборудования. Например, для данных, связанных с циклом У-70, период должен равняться времени цикла

(приблизительно 9 секунд), а для данных по температурным измерениям период может составлять десятки секунд.

Технически создание виртуальной копии оборудования решается с помощью базы данных (БД), расположенной на сервере. Виртуальной копией состояния технологического оборудования является *таблица состояния*. Поля таблицы состояния соответствуют параметрам оборудования, а строки – элементам оборудования. Например, для источников питания таблица состояния содержит следующие поля (приведен не весь список полей):

- IDMOE – идентификатор ИП
- curR - считанное значение напряжения
- onoffR - считанное значение переключателя ЦАП/шунт
- errCurR - код выполнения последней операции полинга
- errInit - код выполнения последней операции инициализации контроллера ИП

Драйвер производит постоянный периодический полинг оборудования и считанные данные (вместе с кодами ошибок) записываются в таблицы. Пользователь в любой момент времени может сделать выборку из базы данных и просмотреть любую интересующую его информацию.

Механизм управления оборудованием с помощью БД реализуется через *управляющую таблицу* и заключается в следующем. Оператор записывает в поля управляющей таблицы инструкции для КПК, а КПК периодически опрашивает эти поля и выполняет записанные в них инструкции. Результат выполнения (данные и код ошибки) он записывает обратно в БД. Например, для источников питания таблица управления содержит следующие поля (приведен не весь список полей):

- IDMOE – идентификатор ИП
- curW - величина уставки напряжения, записанной в ИП
- errCurW - код выполнения последней операции записи уставки в ИП
- ScurW – флаг старта передачи уставки напряжения в ИП
- RcurW – флаг завершения передачи уставки напряжения в ИП
- onoffW – заданное значение переключателя ЦАП/шунт
- SonoffW – флаг старта передачи уставки переключателя ЦАП/шунт

– Ronoff W– флаг завершения передачи уставки переключателя ЦАП/шунт

Для использования возможностей СУБД по разделению прав пользователей, прямой доступ в таблицы запрещается, а для работы с таблицами используются хранимые процедуры.

5. Описание алгоритма управления

Алгоритм управления с использованием управляющей таблицы работает следующим образом. В базе данных существует таблица TABLE, каждая строка которой соответствует управляемому параметру оборудования. Столбцы таблицы:

Value	значение, которое надо передать в оборудование
Start	флаг старта команды
Ready	флаг выполнения команды
Error	код ошибки

Действия клиента при передаче значения в оборудование

Для передачи значения (например, 5.127) в оборудование, оператор вызывает хранимую процедуру ***cliPutCmdWrite*** со следующим содержимым:

```
UPDATE TABLE SET VALUE= 5.127, START = 1, READY = 0
```

Назначение этой процедуры – информировать КПК о желании оператора управлять оборудованием (т.е. передать в оборудование величину 5.127). Далее клиент должен ожидать завершения выполнения команды, для чего он периодически вызывает процедуру ***cliGetCmdWrite*** со следующим содержимым:

```
SELECT READY, ERROR FROM TABLE
```

Если значение READY стало равным 1, то команда выполнена и значение ERROR дает код ошибки или 0, если ошибки не было.

Действия КПК при передаче значения в оборудование

КПК периодически вызывает процедуру ***svrGetCmdWrite*** с целью проверки наличия команды от оператора. Текст процедуры:

SELECT VALUE, START FROM TABLE

Обнаружив, что флаг START=1, КПК его сбрасывает и принимается выполнять операцию записи значения VALUE в оборудование. Выполнив операцию, КПК вызывает процедуру ***svrPutCmdWrite***, в которой он информирует оператора о завершении операции:

UPDATE TABLE SET ERROR=0, READY = 1

Схема совместной работы оператора и КПК при осуществлении операции передачи данных в оборудование приведена в Табл. 1:

Таблица 1. Схема передачи данных в оборудование

Оператор	*БД*	*КПК*
cliPutCmdWrite	Start = 1 Ready = 0 Value=5.127	svrGetCmdWrite
	Start = 0	... выполнение команды ...
	Ready = 1 Error = код результата	svrPutCmdWrite
cliGetCmdWrite		

Схема чтения данных из оборудования аналогична, отличие состоит только в том, что значение Value не записывается в оборудование, а считывается из него.

При использовании изложенного метода технический результат обеспечивается за счет обмена данными через промежуточную таблицу и использование механизма хранимых процедур. Возможность множественного управления

обеспечивается «неделимостью» операций cliPutCmdWrite и fecPutCmdWrite, что также обеспечивается механизмом транзакций.

Заключение

Использование описанного в работе метода управления оборудованием с использованием базы данных обеспечивает простоту и быстроту разработки систем управления оборудованием по сравнению с традиционными общепринятыми способами. СУБД должна удовлетворять следующим условиям:

1. Соответствие стандарту SQL/92.
2. Возможность использования хранимых процедур.
3. Возможность организации множественного удаленного доступа к хранимым процедурам.
4. Наличие транзакций и откатов.

Описанный выше метод был применен для создания системы управления магнитооптическими элементами каналов вывода частиц [1]. Используется СУБД MS SQL Server Enterprise 2008 [2]. Общее число управляемых единиц оборудования составило 132. Список удаленных клиентов, которым разрешено управление источниками питания, приведен в Табл. 2:

Таблица 2. Список удаленных клиентов, управляющих ИП МОЭ

№	Расположение	Управляемые МОЭ	Кол-во клиентов
1	Установка ВЕС (зд.1БВ)	элементы 4-го канала	1
2	Установка СПИН (зд.1БВ)	канал 8, элементы 1с-7с	1
3	Установка ГИПЕРОН (зд.1БВ)	элементы 18-го канала	1
4	Центральный пульт каналов (зд. ВП1)	элементы всех каналов	3
5	Пульт управления ИП (зд.10)	элементы всех каналов	1
6	Установка ФОДС	магнит установки	1

При этом количество удаленных клиентов, которым разрешен только просмотр текущего состояния МОЭ, практически неограничен.

Список литературы

[1] В.Н.Алфёров, И.В.Лобов, А.В.Лутчев, Ю.В.Бордановский, В.С.Лагутин, Д.Г. Хмарук. Система управления источниками питания магнитооптических элементов каналов частиц. Множественный доступ и представление данных. Препринт ИФВЭ 2011-3, Протвино, 2011.

[2] Роберт Э.Уолтерс, Майкл Коулс. SQL Server 2008: ускоренный курс для профессионалов (Accelerated SQL Server 2008). – М.: «Вильямс», 2008.

Рукопись поступила 25 октября 2011 г.

И.В. Лобов, А.В. Лутчев
Управление технологическим оборудованием с использованием реляционной базы данных.

Препринт отпечатан с оригинала-макета, подготовленного авторами.

Подписано к печати 01.11.2011. Формат 60 × 84/16. Офсетная печать.
Печ. л. 1,12. Уч.- изд. л. 1,73. Тираж 80. Заказ 24. Индекс 3649.

ГНЦ РФ Институт физики высоких энергий
142281, Протвино Московской обл.

Индекс 3649

П Р Е П Р И Н Т 2011-26, ИФВЭ, 2011