УДК 004.051

Организация взаимодействия между базами данных предприятий аэрокосмической отрасли в рамках автоматизированного учета с использованием ERP-системы.

А.Ю. Арженко, П.А. Волков, В.А. Вестяк

Аннотация

В силу большого количества участников процесса создания авиационно-космической техники одной из важнейших задач является организация взаимодействия между базами данных, обеспечивающими учет деятельности предприятий аэрокосмической отрасли на каждом этапе производства - от закупки материалов до реализации готовой продукции - в процессе создания разветвленной системы автоматизированного учета на основе автоматизированной системы управления ресурсами предприятия ERP-класса. В целях повышения эффективности автоматизации ранее авторами было предложено создание такой системы и разработка объектов обмена данными. В результате внедрения таких объектов с учетом требования конкретного производства возникает принципиально новая система автоматического управления (САУ) – система автоматизированного учета производства, позволяющая наиболее эффективно использовать информационно-вычислительные ресурсы предприятия при производстве летательных аппаратов, а также иной авиационной и космической техники.

Ключевые слова

ERP-система; эффективность; автоматизация; управление производством; база данных

Одним из решающих факторов эффективного и безопасного производства авиационно-космической техники в современной авиационной и космической промышленности является надежность и мощность системы автоматизированного учета производственных процессов. Ключевым моментом, определяющим степень эффективности автоматизации, является объединение в единый информационный поток данных, отражающих деятельность подраз-

делений предприятия на всех этапах его функционирования: закупка материалов, необходимых для производства летательных аппаратов, хранение этих материалов на складах и их поступление в сборочные цеха, контроль качества изготовленной продукции, поставка изготовленной продукции клиентам компании, а также учет финансовой и хозяйственной деятельности аэрокосмических компании. Объединение данных в единый информационный поток возможен при использовании в создании системы автоматизации деятельности предприятия ERP-систем. В качестве решения повышения эффективности функционирования ERP-систем при увеличении нагрузки в процессе развития производства, не требующего смены ERP-системы и внедрения дополнительного программного обеспечения, нами предлагалось создание нескольких взаимосвязанных баз данных (БД) вместо единой базы, каждая из которых может быть расположена на отдельном сервере и отвечать за определенную сферу деятельности предприятия [1]. Таким образом, нагрузка на систему в целом перераспределяется на несколько серверов. Это позволяет устранить проблему снижения скорости функционирования ERP-системы, что особенно важно при построении САУ в производстве авиационной и космической техники.

Однако создание нескольких БД еще не является решением задачи по перераспределению нагрузки на систему автоматизированного учета. Если базы смогут функционировать лишь автономно, без возможности обмена данными между собой, будут утрачена возможность ведения общих данных по деятельности компании на всех этапах производства летательных аппаратов. Ключевым моментом в реализации предлагаемого метода повышения эффективности автоматизации производственного учета является настройка взаимодействия между базами посредством создания объектов обмена данными, которая будет рассмотрена в данной статье на примере ERP-системы Microsoft Dynamics NAV. Эта система пользуется широким спросом на российском рынке ERP-систем ввиду своей надежности и эффективности. Система обладает гибкой средой разработки, позволяющей как доработку базовых модулей, так и создание новых функционалов, необходимых для автоматизации специфического производства конкретного предприятия [2]. По своей функциональности Microsoft Dynamics NAV является классической ERP-системой, которую возможно настроить с учетом требований производства в любой области, в том числе в авиационной и космической промышленности. Это позволяет использовать Microsoft Dynamics NAV в качестве примера использования предлагаемой концепции в реальном производстве.

При реализации предлагаемого метода повышения эффективности автоматизации учета происходит создание новых БД в системе. При создании отдельных БД, вероятнее всего, одна из баз, в которой ведется основной учет по деятельности компании, будет основной,

а все другие БД, отвечающие за автоматизацию деятельности компании в отдельных областях, будут вспомогательными. Для полноценного функционирования такой системы необходимо обеспечить полный обмен данными между основной и вспомогательными БД.

Базовым средством обмена информацией в системе Microsoft Dynamics NAV являются порты данных. Это объекты, используемые для импорта данных во внешние текстовые файлы и экспорта данных из этих файлов. Существуют широкие возможности выбора при определении формата внешнего файла в процессе импорта и экспорта. Порты данных могут быть динамическими, т.е. при выполнении порт данных определяет, является ли процесс импортом или экспортом. Это достигается посредством опций, которые определяются в форме пользовательского запроса или путем программирования.

При создании второстепенных БД, изначально существующая информация из таблиц-справочников основной базы, которая должна быть общей для основной и вспомогательных БД, может быть перенесена во вспомогательные базы единовременно именно с помощью портов данных.

Однако под объектами обмена данными между базами в рамках предлагаемого метода подразумеваются объекты, доступные пользователям, имеющие пользовательский интерфейс и позволяющие совершать обмен постоянно обновляющейся в процессе учета информацией. Для такой цели порты данных не годятся. Ведь для того, чтобы перенести данные из одной базы в другую с их помощью, пользователю потребуется запустить порт данных, выбрать путь для файла, выгрузить информацию в этот файл, затем запустить порт данных в другой базе, указать путь к файлу, в который были выгружены данные, и выполнить загрузку. Такой алгоритм действий слишком сложен для частого применения и требует временных затрат. Поэтому порты данных хороши лишь при периодическом пакетном обмене информацией. Для эффективной организации взаимодействия между БД потребуется другое решение, освобождающее пользователей от многократных механических действий.

Для реализации обмена информацией между базами предлагается прибегнуть к средствам управления БД, с которой связана ERP-система. Система Microsoft Dynamics NAV может оперировать с данными в двух режимах: используя собственную БД или используя СУБД Microsoft SQL. Собственная БД имеет ограничение по объему, и позволяет хранить крайне мало информации, поэтому для серьезного производства она не годится, и для автоматизации деятельности реального предприятия, как правило, используется СУБД Microsoft SQL, с которой связана ERP-система NAV. Любая современная ERP-система обладает достаточно гибкой средой разработки, подразумевающей возможность интеграции с другими программными средствами и использования их методов [3]. Таким образом, имеется возмож-

ность создания объектов обмена данными с использованием средств Microsoft SQL. Созданные таким образом объекты позволят производить единовременную выгрузку данных из одной БД и загрузку этих данных в другую одним нажатием кнопки. Именно такие объекты должны быть разработаны для обмена информацией между БД системы.

Для обеспечения полного обмена данными разработчику необходимо создать объекты, обеспечивающие возможность загрузки во вспомогательные базы из основной периодически обновляющейся справочной информации по объектам учета (товарам, ресурсам, материалам и т.д.) и возможность загрузки из вспомогательных баз в основную новых данных по учтенным во вспомогательных базах операциям.

Для удобства разработки объектов обмена данными, использующих методы SQL Server, в каждой базе системы предлагается создать программный модуль с именем ADO (от англ. ActiveX Data Objects — «объекты данных ActiveX»). Программный модуль можно представить в виде контейнера для кода, который программист собирается использовать во многих прикладных объектах. Модуль ADO предназначен для хранения разработанных функций, которые обеспечивают проведение основных операций с данными в SQL Server. Далее описываются эти функции. В скобках указаны параметры, использующиеся при вызове функции.

1. ADOConnect(ConnectionString). Назначение данной функции — создание соединения для доступа к объектам данных. Параметр ConnectionString является текстовой строкой, в которую должен передаваться путь к объекту, к которому требуется установить подключение. Код функции:

```
CREATE(adoCon);
adoCon.Open(ConnectionString);
adoCon.CommandTimeout := 60;
IF  adoCon.State = 1 THEN
  SUCCESS := TRUE
ELSE SUCCESS := FALSE;
```

С помощью функции среды ERP-системы CREATE посредством переменной adoCon инициализируется семейство объектов Connection. Переменная adoCon объявляется в списке глобальных переменных модуля, т.е. эта переменная доступна для вызова из любой функции модуля. При объявлении этой переменной ей был присвоен тип Automation. С помощью переменных такого типа возможен вызов методов и функций внешних библиотек и программ-

ных сред. Подтип этой переменной 'Microsoft ActiveX Data Objects 2.6 Library'.Connection, т.е. через эту переменную возможен вызов объектов семейства Connection интерфейса программирования приложений Microsoft ActiveX. Данное семейство объектов обеспечивает подключение к удаленным источникам данных.

После инициализации группы объектов подключения к удаленным источникам данных вызывается объект Open, в который передается в качестве параметра текстовая строка с путем к источнику данных. Метод Open производит открытие объекта данных.

С помощью свойства CommandTimeout, во избежание зависания подключения, указывается в секундах время, в течение которого должно выполняться подключение до сброса выполнения команды.

Булева переменная SUCCESS является возвращаемым значением функции ADOConnect, зависящим от результата попытки подключения к удаленному источнику данных. Свойство State возвращает статус подключения. Если получено значение 1 (подключение установлено), функция ADOConnect возвращает значение TRUE, в противном случае — FALSE.

2. ADOOpenRecordset(CommandText). Назначение данной функции — получение данных из источника, к которому было установлено подключение с помощью функции ADOConnect. Параметр CommandText представляет собой массив текстовых строк из 15 элементов, в который передается строка-источник запроса, в данном случае, SQL-выражение. Код функции:

```
CREATE(adoRst);
adoRst.Open(CommandText[1] + CommandText[2] + CommandText[3] + CommandText[4] + CommandText[5] + CommandText[6] + CommandText[7] + CommandText[8] + CommandText[9] + CommandText[10] + CommandText[11] + CommandText[12] + CommandText[13] + CommandText[14] + CommandText[15] , adoCon);
IF  adoRst.State = 1 THEN
  SUCCESS := TRUE
ELSE SUCCESS := FALSE;
```

С помощью функции среды ERP-системы CREATE посредством переменной adoRst инициализируется семейство объектов RecordSet. Глобальная переменная adoRst является переменной типа Automation, которой присвоен подтип 'Microsoft ActiveX Data Objects 2.6 Library'.Recordset. Посредством этой переменной возможен вызов объектов семейства Re-

cordset интерфейса программирования приложений Microsoft ActiveX. Данное семейство объектов обеспечивает управление набором строк, полученных из источника данных.

Далее происходит вызов функции Open инициализированной библиотеки Recordset, с помощью которой производится получение данных из требуемого источника в соответствии с запросом, полученным в массиве строк CommandText при запуске функции ADOOpenRecordset. Массив передается в функцию в качестве первого параметра единой строкой, составленной из поочередно присоединенных друг к другу элементов. В качестве второго параметра, указывающего на источник, из которого необходимо получить данные, в функцию Open передается значение глобальной переменной adoCon типа 'Microsoft ActiveX Data Objects 2.6 Library'.Connection, инициализированной в функции ADOConnect. Таким образом, получаются строки с данными из заранее открытого источника.

Булева переменная SUCCESS, являющаяся возвращаемым значением функции ADOOpenRecordset, возвращает значение TRUE в случае удачного получения записей из источника и значение FALSE — в обратном случае.

3. ADOMoveFirst(). Данная функция используется для взятия первой по порядку записи в полученном с помощью функции ADOOpenRecordset наборе строк. Код функции:

```
adoRst.MoveFirst();
IF ((adoRst.BOF() = TRUE) OR (adoRst.EOF() = TRUE)) THEN EXIT (FALSE);
SUCCESS := TRUE;
```

Используя метод MoveFirst семейства RecordSet, курсор перемещается на первую строку набора записей, полученных из удаленного источника. В RecordSet перед первой записью, полученной из источника, помещается специальная запись BOF (от Begin Of File), после последней записи находится специальная запись EOF (End Of File). На всякий случай проверяется, не находится ли курсор в этих позициях, в случае чего функция возвращает значение FALSE. Если же все в порядке, и первая запись получена, функция возвращает значение TRUE.

4. ADONextRecord(). Данная функция обеспечивает переход к следующей строке в наборе записей, полученных из источника данных. Код функции:

```
adoRst.MoveNext();
IF adoRst.EOF() = TRUE THEN BEGIN
  adoRst.MoveLast();
  EXIT (FALSE);
END;
```

```
SUCCESS := TRUE;
```

При помощи метода MoveNext семейства RecordSet производится попытка перемещения курсора на следующую запись. Если не была найдена следующая запись, функция возвращает значение FALSE, в противном случае — TRUE.

5. ADOFieldValue(FieldName). Данная функция используется для получения значения поля в строке источника данных в текстовом формате данных. В функцию передается текстовый параметр FieldName с именем поля, значение которого необходимо получить. Код функции:

```
adoFld := adoRst.Fields().Item(FieldName);
TmpValue := adoFld.Value;
FieldValue := FORMAT(TmpValue)
```

Набор Fields из семейства RecordSet содержит объекты, отвечающие за действия с полями источника записей. Одним из таких объектов является метод Item, с помощью которого возвращается конкретное поле, имя которого передается в качестве параметра объекта Item. Этот поле присваивается глобальной переменной adoFld типа 'Microsoft ActiveX Data Objects 2.6 Library'.Field. С помощью переменной такого типа происходит вызов объектов семейства Field библиотеки 'Microsoft ActiveX. После инициализации переменной adoFld при помощи метода Value значение поля передается переменной TmpValue типа Variant. Переменные такого типа могут принимать значения в любом формате данных.

Функция программной среды ERP-системы FORMAT производит форматирование значения переменной в соответствии с передаваемыми в нее параметрами. Если в эту функцию передается только первый параметр, представляющий собой само форматируемое значение, то функция преобразует это значение в текстовую строку без изменений. Итак, с помощью функции FORMAT возвращаемой в переменную FieldValue, являющуюся возвращаемым значением функции, передается полученное значение поля источника записи, преобразованное в текстовую строку.

6. Функции ADOGetDecimalFieldValue(FieldName), ADOGetIntFieldValue(FieldName), и ADOGetDateFieldValue(FieldName). Эти функции предназначены для получения значения поля источника записи с типом данных десятичным значением, целочисленным значением и значением типа «Дата» соответственно. Текстовый параметр FieldName представляет собой имя поля. Код у всех трех функций одинаков:

```
adoFld := adoRst.Fields().Item(FieldName);
RETURN := adoFld.Value;
```

Различие между функциями состоит в том, что возвращаемое значение RETURN в функции ADOGetDecimalFieldValue является десятичным, в функции ADOGetIntFieldValue — целочисленным, а в функции ADOGetDateFieldValue — датой.

7. ADODispose(). Назначение этой функции — закрытие полученного ранее набора строк данных и соединения с удаленным источником данных, а также очистка инициализированных переменных типа Microsoft ActiveX. Эту функцию следует вызывать по завершении работы с удаленным источником данных для того, чтобы прервать функционирование ранее вызванных объектов. Код функции:

```
IF NOT ISCLEAR(adoRst) THEN
  IF adoRst.State = 1 THEN
    adoRst.Close;
IF NOT ISCLEAR(adoCon) THEN
  IF adoCon.State = 1 THEN
    adoCon.Close;
CLEAR(adoFld);
CLEAR(adoRst);
CLEAR(adoCon);
```

Если ранее был получен набор записей из удаленного источника, он удаляется из памяти системы. Если ранее было установлено соединение с удаленным источником данных, оно закрывается. Затем очищаются переменные типа Microsoft ActiveX.

После создания программного модуля с функциями, отвечающими за вызов методов Microsoft SQL Server, можно создавать объекты, обеспечивающие обмен данными между базами системы и использующие функции, описанные в программном модуле.

В качестве примера можно рассмотреть объект, расположенный в основной БД и отвечающий за загрузку данных по учтенным счетам продажи из второстепенной БД некоего предприятия. Данный объект будет являться отчетом, так как отчеты в Microsoft Dynamics NAV удобны не только для вывода информации, но и в качестве функциональных объектов, использующихся для выполнения заданий, вызываемых пользователями.

Прежде чем приступить к описанию объекта, необходимо пояснить, что представляет собой документ в Microsoft Dynamics NAV. Как и во многих других ERP-системах, в Microsoft Dynamics NAV документ, представляющий собой объект учета, состоит из заголовка и строк. Под заголовком подразумевается запись из таблицы, содержащей общую информацию по документу, такую как его номер, код клиента или поставщика по справочнику, дата учета документа, и т.д. Документ может быть описан только одним заголовком. Строки документа представляют собой связанные по ключевым полям с заголовком записи из таблицы, содержащей данные по описываемым в документе операциям, такие как номер товара, его стоимость и количество и т.д. Для каждого документа может быть сколь угодно много строк [4].

Описываемый отчет будет вызываться с использованием трех параметров: даты, за которую требуется загрузить операции, тип загружаемых операций (учтенные или неучтенные) и тип формируемой строки (по каждой операции в отдельности или сгруппированные по ряду полей данные).

Алгоритм работы отчета следующий. При запуске отчет выполняет подключение к удаленной второстепенной БД с помощью функции ADOConnect, в параметр которой передается текстовая строка с адресом этой БД. Далее, в зависимости от выбранного типа загружаемых операций, формируется текстовая строка с запросом, описывающим обращение к заголовкам либо учтенных, либо неучтенных клиентских счетов во второстепенной БД за указанную в параметрах пользователем дату. Далее этот запрос передается в функцию ADOOpenRecordset, посредством которой производится получение строк с данными из удаленного источника, в соответствии с переданным запросом. Затем при помощи функции ADOMoveFirst курсор перемещается на первую запись. После установки курсора на первой записи производится запуск цикла, выполняемого до тех пор, пока функция ADONextRecord возвращает значение TRUE, то есть находится следующая запись в полученном наборе.

В теле цикла происходит инициализация новой записи в таблице заголовков неучтенных счетов (так как все операции из других БД загружаются в основную базу как неучтенные), затем полям инициализированной записи присваиваются соответствующие значения полей текущей строки из набора полученных строк, после чего происходит вставка новой записи в таблицу. Далее формируется запрос, описывающий выборку в удаленной БД записей из таблицы строк, отфильтрованной по номеру документа из строки заголовка. Причем, в зависимости от выбранного пользователем значения параметра отчета «тип формируемой строки», запрос либо описывает взятие каждой строки по документу, либо группировку этих строк по определенному признаку, например, учетная группа товара или клиента. После это-

го производится взятие из БД корпоративных продаж строк документа в соответствии с запросом, создание записей и заполнение их полей в таблице строк неучтенных документов в цикле, отрабатываемом до тех пор, пока находится следующая запись в полученном наборе.

После заполнения заголовка и строк по документу с помощью функции ADONextRecord курсор переходит на следующую запись в наборе строк таблицы заголовков счетов в БД корпоративных продаж, и вновь происходит заполнение заголовка и строк по данному счету.

В результате выполнения отчета происходит импорт данных из вспомогательной БД по учтенным или неучтенным (на выбор пользователя) счетам продажи за указанную дату. Таким образом, описанный объект позволяет производить обмен данными по клиентским продажам между базой головного предприятия и вспомогательной базой, отвечающей за учет деятельности предприятия в определенной сфере деятельности.

Аналогичным образом могут быть созданы и другие объекты, отвечающие за обмен данными между базами. В результате чего будет обеспечена возможность получения всей необходимой информации по учету производства компании в любой момент времени, что и является одним из преимуществ ERP-систем, распределив при этом нагрузку, оказываемую на систему автоматизированного учета между несколькими БД, расположенных на разных серверах, что позволит устранить проблемы снижения быстродействия системы.

Описанный в данной статье объект обмена данными подразумевает ручной запуск и выбор параметров самими пользователями, однако, при необходимости автоматического выполнения обмена данными, эти операции можно описать посредством процедур Microsoft SQL Server, без использования объектов среды ERP-системы и настроить автозапуск этих процедур в определенное время или сразу после совершения операции. Таким образом, весь обмен данными будет происходить в фоновом режиме, а пользователи будут получать уже готовую информацию.

Подводя итоги вышесказанного, можно утверждать, что использование средств среды разработки ERP-системы в сочетании с использованием методов СУБД, с которой эта система интегрирована, предоставляет разработчику широкий спектр возможностей для создания принципиально новой распределенной системы автоматизированного учета производственных процессов, котороая позволит объединить данные учета производственной деятельности всех подразделений предприятия и повысить эффективность автоматизации при разработке САУ в авиационной и космической промышленности, где точность данных и надежность автоматизации производственных процессов являются одними из основных факторов, обусловливающих качество и скорость производства летательных аппаратов, удобство взаимо-

отношений с клиентами и поставщиками, а также эффективную организацию работы персонала.

Библиографический список.

1. Волков П.А. Оптимизация автоматизированной системы учета посредством создания взаимосвязанных баз данных / П.А. Волков //Сборник трудов Международной научно практической конференции «Информационные технологии в образовании, науке и производстве». – 2009 Часть 2. − Серпухов, 2009. − с. 161−163.

2. Дэниел О'Лири. ERP системы. Современное планирование и управление ресурсами предприятия. − М.: Вершина, 2004. − 272с.

3. Попов А. Выбор ERP-системы и подход к автоматизации предприятия − http://www.cfin.ru/itm/kis/choose/questions.shtml, 2009.

4. Грекул В.И., Коровкина Н.Л., Богословцев Д.А., Синайская Н.Н. Автоматизация деятельности предприятия розничной торговли с использованием информационной системы Microsoft Dynamics NAV. − М.: Интернет-университет информационных технологий - ИНТУИТ.ру, 2008. - 184 с.

Сведения об авторах

Арженко Александр Юрьевич, профессор Московского авиационного института (государственного технического университета), д.ф.-м.н.
МАИ, Волоколамское ш., 4, Москва, Ф-80, ГСП-3, 125993;
тел:8 (903) 769-08-27, 497-94-03

Волков Павел Андреевич, аспирант Московского авиационного института. (государственного технического университета), тел: 8 (916) 332-18-78, 8 (499) 199-37-45, E-mail: nuclearfrost@yandex.ru

Вестяк Владимир Анатольевич, заведующий кафедрой Московского авиационного института. (государственного технического университета), к.ф.-м.н., доцент
МАИ, Волоколамское ш., 4, Москва, Ф-80, ГСП-3, 125993;
E-mail: v.a.vestyak@mail.ru