Простые генераторы отчетов для учетно-управленческих информационных систем

А.Б. Бетелин[1], И.Б. Егорычев[2], А.А. Прилипко[3], Г.А. Прилипко[4], С.Г. Романюк[5], Д.В. Самборский[6]

[1]ab@niisi.msk.ru, [2]egorychev@gmail.com, [3]aaprilipko@niisi.msk.ru, [4]prilipko@niisi.msk.ru, [5]sgrom@niisi.ras.ru, [6]samborsky_d@fastmail.com

[1,2,3,4,5,6]ФГУ ФНЦ НИИСИ РАН, Москва, Россия

Аннотация. В статье приводится описание функциональных возможностей и особенностей работы средств генерации отчетов, применяемых в учетно-управленческих информационных системах НИИСИ РАН.

Ключевые слова: генератор отчетов, информационная система, учет, управление, программное обеспечение, база данных

1. Введение

В современном мире, пожалуй, ни одна программная информационная система не может обойтись без генератора отчетов.

В контексте информационных систем отчетом принято считать представление информации, извлеченной из хранилища системы, в сформатированном виде, удобном для визуального восприятия человеком. Отчет может быть отображен на экране монитора, напечатан на принтере, выведен в файл и т.п.

Создание отчетов является одной из важных задач, возникающих при разработке различных информационных систем. Отчет включает, помимо содержательной информации, еще и некоторую оформительскую составляющую, формирование которой при помощи стандартных средств зачастую представляет собой весьма трудоемкую задачу. Поэтому в программных информационных системах решение этой задачи обычно возлагается на генератор отчетов — специальное программное обеспечение, позволяющее представить информацию в удобочитаемом структурированном виде [6]. На основе входных данных генератор формирует документ, который затем можно напечатать на бумаге или же сохранить в файл в каком-либо из стандартных форматов данных, применяемых для электронных документов.

В общем случае источником исходного материала для генератора отчетов могут служить файлы с данными, а также другие программы, программные системы и комплексы, осуществляющие ввод, обработку и хранение информации. В информационных системах данные извлекаются, как правило, из хранилища системы, представляющего собой набор файлов или же базу данных.

В настоящее время существует великое множество разнообразных генераторов отчетов, реализованных как в виде составной части программной системы (встроенные генераторы), так и в виде самостоятельной программы. В последнем случае генератор, как правило, снабжается программным интерфейсом, позволяющим воспользоваться его функциональностью из другой программы или программной системы.

Встроенные генераторы отчетов обычно используются в различных СУБД (например, Access [8], Informix [9]), а также в учетных системах семейства «1С:Предприятие» [7]. Среди самостоятельных программ формирования отчетов широко известны Crystal Reports [10], FastReport [11], Stimulsoft Reports [12]. Кроме упомянутых выше и многих других коммерческих продуктов в последние годы также получили распространение несколько программ генерации отчетов с открытым исходным кодом, например: YARG [14], Carbone [15], appy.pod [16]. Эти программы объединяет то, что они самостоятельно редактируют содержимое XML-данных, содержащихся в документах стандартов OpenDocument (ODF) [17] и Office Open XML (OOXML) [18], и избегают использования сред OpenOffice [19] и Microsoft Office [20].

В НИИСИ РАН в разное время был разработан ряд программных систем для автоматизации управленческого и бухгалтерского учета [1]. Эти системы снабжены средствами генерации отчетов, которые создавались с учетом следующих потребностей и условий:

— пользователем учетных систем является административно-технический персонал организации, не имеющий навыков программирования;

— все отчеты являются типовыми, каждый из них имеет заранее определенный вид (заголовок, наполнение, размещение информации на

печатной странице);

— не требуется создание отчетов в произвольной форме с возможностью индивидуального программирования;

— в отчетах не требуется наличие графиков, диаграмм, иллюстраций и т.п.; информация представляется исключительно в виде текста.

Общая схема функционирования разработанных инструментов создания отчетов состоит в следующем. Система с помощью встроенных средств для взаимодействия с базой данных извлекает из нее нужную для отчета информацию, обрабатывает ее и на этой основе формирует входной поток информации для генератора отчетов. Далее генератор отчетов принимает входной поток и, используя содержащуюся там информацию, преобразует шаблон отчета в итоговый документ, который выводит в файл заданного формата. В зависимости от применяемого генератора это может быть текстовый файл или же документ в формате электронной таблицы или текстового процессора сред OpenOffice и MS Office.

Настоящая статья посвящена описанию функциональных возможностей и особенностей работы рассматриваемых средств генерации отчетов.

2. Генератор отчетов системы БРОД

Система автоматизации бухгалтерского и налогового учета БРОД [1] была создана в НИИСИ РАН более 25 лет назад и продолжает развиваться и эксплуатироваться в комплексе со стандартными бухгалтерскими системами. Пользовательский интерфейс системы ориентирован на взаимодействие в стиле алфавитно-цифрового (а/ц) терминала. Соответственно, в оформлении отчетов применяются символы из имеющихся на клавиатуре а/ц терминала, дополненные набором символов псевдографики.

Использование стандартных средств генерации отчетов, предоставляемых языками 4gl [13], оказалось достаточно трудоемким и даже при небольших изменениях в отчетах каждый раз требовало модификации и пересборки программ. Поэтому более технологичным решением явилось создание специализированного комплекса программ, совместно решающих задачи как формирования отчетов, так и их последующего форматирования и печати. Этот комплекс включает программу генерации отчетов **tgf**, обеспечивающую получение текстовых отчетов (краткий обзор tgf приведен в [3]) и набор вспомогательных программ для дополнительной обработки файлов с отчетами.

2.1. Схема работы генератора и команды преобразования шаблона

Шаблон отчета находится в так называемом текстовом файле-форме, обычно с расширением .tgf. Он содержит образец внешнего вида отчета со специальными дополнительными полями, заключенными в квадратные скобки. В них находятся управляющие команды для генератора отчетов, которые могут задавать такие действия как перемещение по файлу-форме, удаление строк из входного потока, позиционирование их в шаблоне и т.п. Генератор считывает содержимое файла-формы и данные из входного потока, а затем, руководствуясь командами в управляющих полях, размещает считанные данные в шаблоне.

Запустить генератор из командной строки ОС Linux [22] можно, например, так:

```
tgf -f=template.tgf - -
```

Здесь template.tgf — имя файла-формы, входной поток поступает со стандартного ввода, а готовый отчет направляется на стандартный вывод.

Управляющие поля в шаблоне могут относиться к одному из двух типов.

Поле первого типа начинается с некоторого числа, за которым следуют одна или несколько однобуквенных команд:

[*число команда...команда*]

Число интерпретируется в зависимости от указанных за ним команд. Оно может обозначать либо номер строки из входного потока (строки нумеруются с 1), либо количество строк, которые надо удалить из входного потока, либо указание, на сколько строк надо переместиться по форме. Например:

l — удалить указанное число строк из начала входного потока;

f — переместиться вверх по файлу-форме на указанное число строк.

Это позволяет организовать циклы при выводе информации. Выход из цикла происходит либо при завершении поступления строк из входного потока, либо если первая строка при выполнении команды f начинается с указанного в последнем условном переходе проверочного слова-признака.

Помимо команд l и f генератор различает также следующие однобуквенные команды:

b — выводить строку с первого непустого символа;

e — вывести продолжение строки;

g — считать квадратную скобку входящей в поле для вывода информации;

n — не выводить данную строку в генерируемый отчет, если содержимое всех полей данной строки пусто;

c — разместить содержимое в центре поля (иначе содержимое прижимается к соответствующему краю поля).

Вот несколько примеров управляющих полей первого типа:

[2bg] — вставить 2-ю строку входного потока, предварительно удалив из нее передние пробелы, и выровнять ее по позиции левой квадратной скобки.

[2eng] — вставить продолжение 2-й строки, причем символ n внутри квадратных скобок указывает, что если продолжения нет, то эту строку нужно будет убрать из получаемого отчета. Таким образом можно создать отчет с числом строк, достаточным для того, чтобы целиком поместить в него строку из входного потока.

[5l] — удалить из входного потока первые пять строк.

[4f] — переместиться вверх по файлу-форме на четыре строки, если не встретилось слово-признак условного перехода.

Поля второго типа служат для реализации условных переходов внутри файла-формы. Такое поле содержит восклицательный знак, перед которым может быть указано число, и некоторое слово-признак, которое запоминается для дальнейшего использования командой f:

[*число*!*слово*]

Если первая строка входного потока начинается с указанного слова *«слово»*, то это вызовет перемещение вниз по файлу-форме на указанное *число* строк.

Рассмотрим следующий пример. Пусть в файле-форме последовательно встречаются управляющие поля [3!КонеЦ], [2l] и [1f] (см. рис. 1), и первая строка входного потока начинается со слова «КонеЦ».

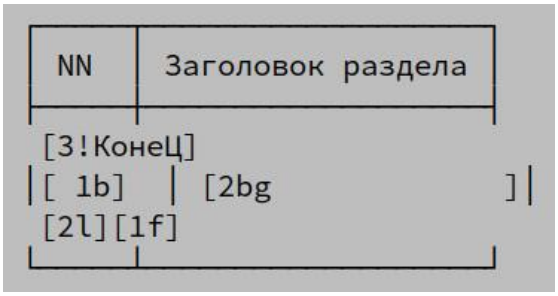

Рис. 1. Пример файла-формы

Тогда в процессе обработки поля [3!КонеЦ] выполняется перемещение вниз на три строки файла-формы с запоминанием слова «КонеЦ». Обработка поля [2l] приведет к удалению первых двух строк из входного потока. Если после этого первая строка входного потока будет начинаться со слова «КонеЦ», то при обработке поля [1f] произойдет смещение на следующую строку файла-формы, иначе будет выполнен переход вверх на одну строку.

Перечисленных механизмов оказалось вполне достаточно для изготовления текстовых отчетов любой сложности, необходимой при выпуске документов бухгалтерского и налогового учета.

Следует отметить, что с помощью генератора отчетов tgf можно решить и обратную задачу, т.е. извлечь информацию из файла с отчетом по некоторой форме, например, для печати содержимого его полей на специальных бланках. Здесь дополнительно используются возможности, предоставляемые программой pcp (см. подраздел 2.2.) по установке позиции на листе бумаге перед печатью необходимой информации.

2.1. Вспомогательные программы обработки текста

Довольно часто после генерации отчета возникает задача дополнительного форматирования находящихся внутри него таблиц. Таблица в данном случае представляется последовательностью текстовых строк, в которых содержимое одной ячейки таблицы отделяется от содержимого другой ячейки таблицы символом-разделителем (обычно для этого используется псевдографический символ вертикальной черты). Для дополнительной обработки таблиц в отчете был разработан ряд программ, функциональные возможности некоторых из них приведены ниже.

pcp — выполняет перекодировку файлов и/или подготовку их для печати на принтерах EPSON, HP и совместимых с ними.

tmk — выравнивает ширину столбцов таблицы, позволяя уменьшить их ширину до минимально необходимой, а также удалить пустые столбцы.

pff — разбивает отчет на страницы с учетом наличия в нем таблиц, вставляет дополнительные заголовки, осуществляет нумерацию страниц.

tpp — позволяет провести разбиение таблиц на страницы по ширине с возможностью учета границ столбцов и переноса нескольких первых столбцов таблицы на новые страницы.

Для того чтобы автоматически выполнять предварительное разбиение на страницы и дополнительное форматирование перед печатью, используются специальные файлы с расширением .ph, в которых указываются ключи, позволяющие перед печатью отчета вызвать необходимые программы с указанными ключами. Имя такого файла должно совпадать с расширением

файла с отчетом. Например, если файл с отчетом называется «report.txt», то специальный файл для него должен называться «.txt.ph».

Довольно часто при работе с информацией возникает задача обработки таблиц или CSV-файлов, полученных из файлов электронной таблицы MS Office, и переноса необходимой информации в систему или базу данных. Для этих целей была разработана программа tes, которая может выделить информацию из нужных столбцов таблицы, переставить их при необходимости, а также добавить пустые столбцы или столбец, содержащий номера строк исходного файла.

Все вышеназванные программы применяются в НИИСИ РАН с 1997 года. Они существенно облегчили решение целого ряда задач, связанных с разработкой и эксплуатацией системы расчета зарплаты, а также ряда других бухгалтерских и информационных систем. Эти программы входят в состав комплекса на основе текстового редактора **rk** [4,5] и поставляются вместе с ним в виде исходного кода на языке Си.

3. Генератор отчетов АСУ НИИСИ

Информационная система АСУ НИИСИ [1] была разработана в 2007 году на базе инструментальной среды с веб-интерфейсом [2] и в настоящее время продолжает развиваться. На текущий момент в АСУ НИИСИ насчитывается около 30 информационных подсистем для решения разного рода учетно-управленческих задач.

Взаимодействие пользователя с подсистемами АСУ НИИСИ осуществляется посредством графического веб-интерфейса, для которого устройством отображения информации служит графический монитор. Создаваемые в АСУ НИИСИ отчеты имеют внешнее оформление документов полиграфического качества, где возможно применение шрифтов различных гарнитур, кеглей, начертаний и т.п., а также таблиц с ячейками, обрамленными графическими линиями. Применяемый в АСУ НИИСИ генератор отчетов входит в состав ее базового инструментария.

3.1. Схема генерации отчета

Каждому отчету в АСУ НИИСИ соответствует отдельный командный SQL-файл, который содержит программу подготовки входного потока информации для генератора отчетов. Программа извлекает данные из базы данных АСУ и формирует строки входного потока. Система запускает этот SQL-файл на выполнение в начале процесса изготовления отчета.

Входной поток представляет собой последовательность текстовых строк, каждая из которых содержит либо команду, либо данные. Строки с командами начинаются с символа «#», а строки с данными такого префикса не имеют. В строках с командами в случае необходимости после символа «%» может присутствовать комментарий. Входной поток располагается в файле формата CSV.

Шаблоном отчета служит документ в формате ODF (ODS в случае электронной таблицы, или же ODT в случае текстового документа). В шаблоне с помощью переменных вида «#varname» обозначены позиции для вставки данных. Генератор считывает команды и данные из входного файла, в процессе выполнения этих команд преобразует шаблон и размещает данные на позициях, обозначенных переменными.

Отметим следующее различие между программами генерации отчетов системы БРОД и АСУ НИИСИ. В генераторе системы БРОД команды, управляющие процессом формирования отчета, размещаются в шаблоне, а входной поток содержит только данные для отчета. В генераторе АСУ НИИСИ, напротив, входной поток содержит и данные, и команды, шаблон же лишь описывает внешний вид отчета.

Кроме того, в отличие от системы БРОД схема работы генератора отчетов АСУ НИИСИ не предполагает перемещение по шаблону. Формирование отчета и подстановка значений переменных производится при помощи механизма работы с буферами. Буфер — это динамически создаваемое хранилище, в которое можно поместить одну или несколько строк из шаблона. Буфер создается автоматически, когда генератор встречает его имя во входном файле. Имя буфера может быть любым словом, содержащим буквы, цифры и символы подчеркивания.

Обычно процесс преобразования шаблона происходит следующим образом. Строки шаблона копируются в один или несколько буферов. Каждая строка, содержащая переменные, обязательно должна быть скопирована в буфер, чтобы присвоить им значения с помощью специальной операции заполнения буфера (см. подраздел 3.3). Из шаблона удаляются строки, не требующиеся в итоговом документе либо находящиеся не там, где необходимо. После этого содержимое буферов, в т.ч. строки с получившими значение переменными, копируется в шаблон в нужные позиции. Описание команд для выполнения указанных операций представлено в подразделе 3.3.

Готовый отчет выводится в файл в виде ODF-документа. При необходимости он может быть сконвертирован в формат электронной таблицы или текстового процессора MS Office с помощью программных средств OpenOffice.

3.2. Программа генерации отчета

Программа генерации отчета, обрабатываю-

щая шаблон и заполняющая его данными, написана на языке Perl и называется **xmlfilter.pl**. Она получает на входе указанный выше CSV-файл с командами и данными и перерабатывает файл content.xml из ODF-шаблона в аналогичный файл для готового отчета. Согласно спецификации формата OpenDocument [17] в файле content.xml находится содержательная часть ODF-документа. Отчет формируется при помощи специального скрипта, который сначала извлекает файл content.xml из ZIP-архива шаблона, затем запускает программу xmlfilter.pl, а в конце помещает переработанный файл content.xml в итоговый ZIP-архив отчета. Таким образом, в процессе создания ODF-документа отчета не используется ни программа текстового процессора OpenOffice, ни прикладные библиотеки этой программы. Это повышает надежность генератора отчета, т.к. опыт применения штатного способа формирования ODF-документов с помощью пакета SDK OpenOffice показал ненадежность работы этого ПО. Позже эти наблюдения подтвердились свидетельствами независимых разработчиков (например, см. описание истории создания генератора отчетов YARG [14]).

В командной строке ОС Linux программа генерации отчета может быть запущена так:

```
cat example.csv | ./bin/xmlfilter.pl \
--in src/content.xml \
--out res/content.xml
```

Здесь:

example.csv — входной файл в формате CSV;

src/content.xml — файл content.xml из шаблона отчета;

res/content.xml — переработанный файл content.xml для готового отчета.

3.3. Команды преобразования шаблона

Трансформация шаблона отчета в итоговый документ происходит в результате выполнения команд, наиболее важные из которых кратко описаны здесь с указанием формата вызова и примерами использования.

Аргументами команд могут быть номера строк шаблона либо имена буферов. В командах, оперирующих со строками, требуется указывать номер листа электронной таблицы и номер обрабатываемой строки (или же диапазон номеров строк). Эти параметры задаются следующим образом:

[*лист*:]*строка1*[*..строка2*][:o[ld]|n[ew]]

Листы электронной таблицы нумеруются натуральными числами, начиная с 0. Точно так же нумеруются строки шаблона. Если в шаблоне содержится ровно один лист, то его номер можно не указывать. Если шаблон является не электронной таблицей, а текстовым ODT-документом с таблицами, то вместо номера листа указывается номер таблицы. Вот несколько примеров указания номеров строк:

1 — строка с номером 1 (вторая строка шаблона);

2..4 — три строки с номерами 2, 3, 4;

1:0..9 — первые десять строк второго по порядку листа электронной таблицы (или же второй по порядку таблицы в ODT-документе).

Способ вычисления номера строки документа задается при помощи суффиксов «:old» (или «:o») и «:new» (или «:n»). Суффикс «:old» определяет позиции строк до выполнения команд вставки и удаления строк (см. ниже), а суффикс «:new» — после указанных операций. По умолчанию предполагается использование суффикса «:new». Например, выражение «5:old» указывает на позицию строки номер 5 до выполнения операций вставки и удаления.

Перейдем теперь к описанию команд.

«>» — **команда копирования**. Позволяет переписать одну или несколько строк из шаблона в буфер, содержимое буфера в итоговый документ или же содержимое одного буфера в другой. Формат вызова:

 # номера_строк | имя_буфера > номера_строк | имя_буфера

Примеры:

2 > BUF_1 — копируется строка номер 2 в буфер BUF_1;

3..4 > BUF_2 — копируются две строки с номерами 3 и 4 в буфер BUF_2;

BUF_2 > 7..8 — копируется содержимое буфера BUF_2 в две строки с номерами 7 и 8, при этом предыдущее содержимое строк 7 и 8 теряется;

BUF_2 > BUF_3 — копируется содержимое буфера BUF_2 в буфер BUF_3.

«>>» — **команда вставки**. Служит для размещения содержимого буфера в итоговом документе перед указанной строкой. Формат вызова:

 # имя_буфера >> номер_строки

Пример:

BUF_2 >> 2 — содержимое буфера BUF_2 помещается перед строкой номер 2.

«delete» — **команда удаления**. Дает возможность убрать из итогового документа одну или

несколько строк. Формат вызова:

delete *номера_строк*

Примеры:
delete 1 — удаляется строка номер 1;
delete 2..5 — удаляются строки с номерами от 2 до 5.

«fill» — команда заполнения буфера. Позволяет присвоить значение одной или нескольким переменным из шаблона. Каждой переменной может соответствовать одиночное значение (если нужно заполнить одну строку в итоговом документе) или же ряд значений (если нужно заполнить несколько строк). Формат вызова:

fill *имя_буфера*

Значения для присвоения переменным представляются в виде массива данных в CSV-формате, где в качестве разделителя значений разных колонок используется символ табуляции. В таком формате выдает данные клиент СУБД MySQL [21] в пакетном режиме. Если массив не пуст, то он должен состоять как минимум из двух строк. В первой строке помещается заголовок с именами колонок, которые должны совпадать с именами переменных в буфере. В последующих строках размещаются данные, соответствующие указанным в заголовке переменным. Порядок колонок может быть произвольным, допускается наличие лишних данных или же частичное их отсутствие. В простейшем случае в первой строке находится имя переменной, а во второй — ее значение. Для хранения этого массива предусмотрен специальный буфер, называемый *аккумулятором*.

Строки с переменными предварительно копируются из шаблона в буфер с именем *«имя_буфера»*. Команда fill выполнит подстановку значений переменных в буфере, используя данные из аккумулятора. Если аккумулятор содержит несколько строк данных, то для каждой такой строки будет создан и заполнен данными из этой строки отдельный экземпляр всех строк буфера. Например, если в буфере находятся две строки шаблона, а в аккумуляторе — три строки данных, то результатом выполнения команды fill будут шесть строк.

Если для некоторых переменных в строках, находящихся в буфере, в массиве данных не задано значений (нет соответствующих колонок), то значения этих переменных останутся неопределенными.

В массиве данных на любой позиции может использоваться специальное значение COUNT. При выполнении команды fill оно заменяется на номер текущей строки данных в массиве.

После заполнения данными содержимое буфера, как правило, копируется в итоговый документ с помощью команды «>» или «>>».

При работе с ODT-шаблоном имеется своя специфика, состоящая в том, что переменные, описанные вне таблиц, недоступны для обычных команд. Этим переменным можно присвоить значения только при помощи команды fill global, где global — имя особого буфера, обозначающего весь ODT-документ.

«reset» — команда сброса аккумулятора. Служит для очистки аккумулятора от содержащихся в нем данных. Формат вызова:

reset

«continue» — команда дополнения содержимого аккумулятора. Формат вызова:

continue

Иногда бывает удобно выводить массив данных не целиком, а отдельными порциями путем последовательного вызова нескольких операторов SELECT. Для этого предусмотрена команда continue, позволяющая продолжить заполнение аккумулятора очередной порцией данных.

Следующий пример показывает, как представляется во входном потоке массив данных, выведенный двумя порциями:

```
#% первая порция массива данных:
n1    n2    n3
1     1     1
2     4     8
# continue
#% вторая порция массива данных:
n1    n2    n3
3     9     27
```

«default» — команда определения подразумеваемого содержимого аккумулятора. Формат вызова:

default
массив_данных

В некоторых случаях оператор SELECT может выдать пустой массив данных. Тогда аккумулятор тоже будет пуст, последующая команда fill не изменит содержимого буфера, а в итоговом документе окажутся переменные, значение которых не определено. Чтобы этого не произошло, предусмотрена возможность задать содержимое аккумулятора по умолчанию, представляющее собой массив данных, который будет использован командой fill в случае, если аккумулятор

пуст.

В следующем примере приведен фрагмент входного потока, содержащий команду сброса аккумулятора, массив данных, команду определения пустого массива данных в качестве подразумеваемого и, наконец, команду заполнения буфера:

```
# reset
<массив данных, возможно пустой>
# default
n1    n2    n3
# fill BUF
```

Команду сброса аккумулятора (reset) следует выполнить до вывода данных, т.к. в противном случае будет использовано содержимое заполненного ранее аккумулятора, а подразумеваемое содержимое не будет воспринято.

Желательно определить подразумеваемое содержимое для каждого потенциально пустого набора данных, поскольку незаполненные переменные буфера не удаляются автоматически ввиду того, что они впоследствии могут быть заполнены на второй итерации или же командой fill global.

Конечно, проблему пустого аккумулятора можно решить на уровне SQL, заранее определив число строк в выводимом массиве данных, и если он пуст, то не выполнять вывод данных.

«drop» — команда удаления листа электронной таблицы или таблицы из текстового документа. Дает возможность удалить указанный лист электронной таблицы или же таблицу из текстового документа. Формат вызова:

drop *номер_листа_или_таблицы*

«columns» — команда копирования листа электронной таблицы. Служит для копирования атрибутов колонок одного листа электронной таблицы в другой лист. Если необходимо скопировать еще и содержимое строк таблицы, то это придется сделать вручную. Формат вызова:

columns *номер_листа* > *номер_листа*

3.4. Особенности, обусловленные характером работы процессора XML

Сегменты шаблона, содержащие более чем одну пустую строку, в ODF XML заменяются одной специальной строкой, что сбивает нумерацию строк. Поэтому в шаблонах следует использовать не более одной пустой строки для разделения данных или же учитывать это обстоятельство, считая сегменты пустых строк за одну строку.

Имена переменных, занимающих всю ячейку и имеющих тип decimal или date, должны содержать суффиксы _n или _d соответственно. Например: #summa_n, #mdate_d. Это связано с тем, что процессору XML необходимо принудительно выставить тип ячейки.

3.5. Пример работы генератора отчетов

Рассмотрим пример отчета, в котором первые три натуральных числа (1, 2 и 3) выводятся в первой степени, в квадрате и в кубе. ODS-шаблон для этого отчета изображен на рис. 2.

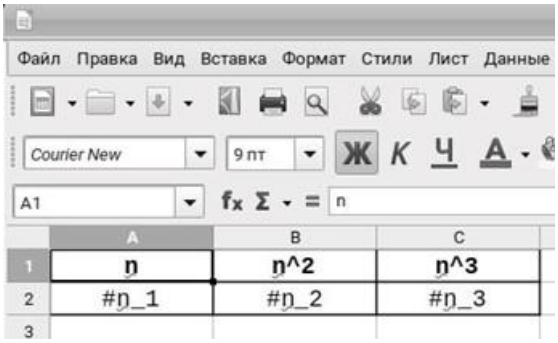

Рис. 2. Пример ODS-шаблона

Первая строка шаблона содержит заголовок отчета, а во второй находятся переменные #n_1, #n_2 и #n_3, которым будут присваиваться значения первой степени, квадрата и куба очередного числа.

Содержимое SQL-файла с программой подготовки входного потока для генератора отчетов может быть, например, таким:

```
SELECT '#% Генерация отчета' AS '#';
SELECT '# 1 > BUF' AS '#';
SELECT '# delete 1' AS '#';
SELECT '#% массив данных:' AS '#';
SELECT i AS 'n_1', i*i AS 'n_2', i*i*i
AS 'n_3' FROM t_num WHERE i <= 3;
SELECT '# fill BUF' AS '#';
SELECT '# BUF >> 1' AS '#';
```

Здесь путем вызова ряда операторов SELECT порождается последовательность команд и данных, образующих входной поток. После выполнения этой SQL-программы входной поток будет помещен в CSV-файл следующего вида:

```
#% Генерация отчета
# 1 > BUF
# delete 1
#% массив данных:
n_1   n_2   n_3
1     1     1
2     4     8
3     9     27
# fill BUF
# BUF >> 1
```

В первой строке файла находится комментарий. Затем идет команда копирования строки с номером 1 в буфер с именем BUF, а после нее — команда удаления строки номер 1 из шаблона. Далее после еще одной строки с комментарием определяется массив данных для присваивания переменным в буфере BUF. В следующей строке выполняется присваивание значений переменным в буфере BUF при помощи команды fill. Наконец, содержимое буфера BUF (переменные, получившие значения) вставляется в шаблон перед строкой номер 1, тем самым формируя итоговый документ.

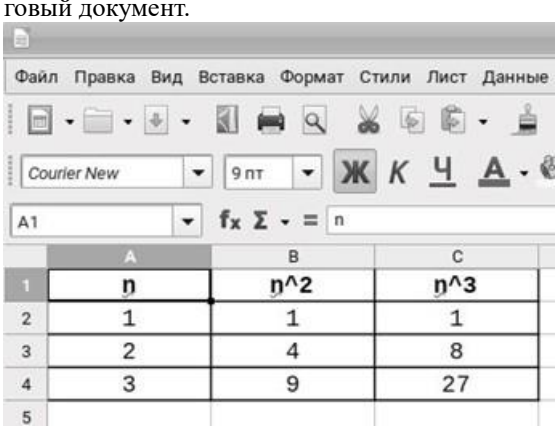

Рис. 3. Пример готового отчета

Далее этот файл отправляется на вход генератора, который в качестве результата своей работы создает файл content.xml для готового отчета в формате ODS, изображенного на рис. 3.

4. Заключение

Описанные в данной работе средства генерации отчетов и вспомогательные программы успешно применяются в НИИСИ РАН не только для создания всевозможных отчетов в учетно-управленческих системах. Они помогают в процессе подготовки отчетности в электронной форме в виде XML-файлов, а также при решении разнообразных задач обработки и преобразования текстовой информации и вопросов миграции данных между информационными системами.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (Проведение фундаментальных научных исследований (47 ГП) по теме «1021060909180-7-1.2.1 Развитие методов математического моделирования распределенных систем и соответствующих методов вычисления. (FNEF-2022-0007)»).

Simple Report Generators for Accounting and Management Information Systems

A.B. Betelin, I.B. Egorychev, A.A. Prilipko, G.A. Prilipko, S.G. Romanyuk, D.V. Samborskiy

Abstract. The article describes the functionality and features of the report generation tools used in accounting and management information systems of SRISA RAS.

Keywords: report generator, information system, accounting, management, software, database

Литература

1. А.Б. Бетелин, И.Б. Егорычев, А.А. Прилипко, Г.А. Прилипко, С.Г. Романюк, Д.В. Самборский. Методы создания распределенного комплекса информационных систем для автоматизации управленческого и бухгалтерского учета. "Труды НИИСИ РАН", т.8 (2018), №4, 175-180.

2. И.Б. Егорычев. Инструментарий для построения автоматизированных учетных систем с WEB-интерфейсом. «Математическое и компьютерное моделирование систем: теоретические и прикладные аспекты», сб. науч. тр. НИИСИ РАН под ред. акад. В.Б.Бетелина. М., НИИСИ РАН, 2009, 81-90.

3. Г.А. Прилипко. Генератор отчетов и обработка текстовой информации. Достижения и приложения современной информатики, математики и физики. "Материалы Всероссийской научно-технической конференции", Уфа, РИЦ БашГу, 2012, 69-71.

4. Г.А. Прилипко. Исследование и разработка интерактивных языковоориентированных систем редактирования и визуализации программ. Автореферат кандидатской диссертации. МГУ им. М.В. Ломоносова, Москва, 1986, 23 с.

5. Г.А. Прилипко. Применение ЭВМ в учебном процессе. Опыт использования ЭВМ в обучении. "Межвузовский сборник научных трудов" под ред. О.М.Петрова. М., ВЗМИ, 1986. 26-30.

6. Генератор отчетов. https://ru.wikipedia.org/wiki/Генератор_отчетов (дата обращения 10.02.2023)

7. Отчеты в 1С. https://www.1s-up.ru/otchety-v-1s/ (дата обращения 10.02.2023)
8. Создание отчетов в Access. https://maxfad.ru/ofis/ms-access/462-sozdanie-otchetov-v-access.html (дата обращения 10.02.2023)
9. Язык программирования баз данных Informix-4GL. https://ami.nstu.ru/~vms/method5/posobie4_4gl.HTM (дата обращения 10.02.2023)
10. Сайт Crystal Reports. https://www.sap.com/products/technology-platform/crystal-reports.html (дата обращения 10.02.2023)
11. Сайт Fast Report. https://быстрыеотчеты.рф/ru/ (дата обращения 10.02.2023)
12. Сайт Stimulsoft Reports. https://www.stimulsoft.com/ru (дата обращения 10.02.2023)
13. Fourth-generation programming language. https://en.wikipedia.org/wiki/Fourth-generation_programming_language (дата обращения 10.02.2023)
14. YARG - open-source библиотека для генерации отчетов. https://www.jmix.ru/cuba-blog/report-generator/ (дата обращения 10.02.2023)
15. Сайт проекта Carbone. https://carbone.io/documentation (дата обращения 10.02.2023)
16. Сайт проекта appy.pod. https://appyframe.work/13 (дата обращения 10.02.2023)
17. OpenDocument technical specification. https://en.wikipedia.org/wiki/OpenDocument_technical_specification (дата обращения 10.02.2023)
18. Office Open XML. https://en.wikipedia.org/wiki/Office_Open_XML (дата обращения 10.02.2023)
19. OpenOffice.org. https://en.wikipedia.org/wiki/OpenOffice.org (дата обращения 10.02.2023)
20. Microsoft Office. https://en.wikipedia.org/wiki/Microsoft_Office (дата обращения 10.02.2023)
21. MySQL. https://en.wikipedia.org/wiki/MySQL (дата обращения 10.02.2023)
22. Linux. https://en.wikipedia.org/wiki/Linux (дата обращения 10.02.2023)