Методика загрузки информации в 1С

Оглавление

[Аннотация 1](#_Toc474877840)

[Введение 1](#_Toc474877841)

[Обзор технологий загрузки данных в 1С 2](#_Toc474877842)

[Импорт внешних файлов 3](#_Toc474877843)

[Импорт текстовых файлов 3](#_Toc474877844)

[Импорт XML-файлов 6](#_Toc474877845)

[Импорт JSON-файлов 10](#_Toc474877846)

[Передача данных через WEB-сервисы 1С 11](#_Toc474877847)

[Использование встроенного механизма обмена данными 12](#_Toc474877848)

[Использование встроенного механизма доступа к внешним ODBC-данным 12](#_Toc474877849)

[Вызов в 1С-скрипте пользовательской внешней компоненты 13](#_Toc474877850)

[Использование в 1С-скрипте COM-объектов 14](#_Toc474877851)

[Вызов пользовательского COM-объекта 14](#_Toc474877852)

[Вызов COM-объекта ADO 15](#_Toc474877853)

[Вызов 1С в качестве COM-сервера 16](#_Toc474877854)

[Прямая загрузка в базу данных 1С 17](#_Toc474877855)

[Общее решение 17](#_Toc474877856)

[Оптимизация для Microsoft SQL Server 18](#_Toc474877857)

[Оптимизация для Oracle 19](#_Toc474877858)

[Выводы 19](#_Toc474877859)

[Ссылки 19](#_Toc474877860)

# Аннотация

В настоящем документе рассмотрены различные способы загрузки информации в 1С из внешних программ (баз данных). Основной акцент делается на наиболее русорсоемком режиме - одномоментной загрузке большого объема данных, например, при переводе информационной структуры предприятия на платформу 1С.

# Введение

При загрузке больших объемов данных из одной системы в другую приходится решать различные по своей сути технические задачи:

* преобразование исходной модели данных к требуемой системой-получателем;
* выбор или разработка технологий загрузки, приемлемых с точки зрения необходимых ресурсов (время загрузки, необходимая память);
* верификация результатов загрузки.

С учетом особенностей платформы 1С (в частности, ее ограничений в части быстродействия и гибкости встроенного языка сценариев) наиболее эффективной будет схема, в которой преобразование исходной модели данных выполняется на основе специального «подготовительного» программного обеспечения, а верификация результатов загрузки, включая возможные вспомогательные обработки типа сведения остатков, - встроенными средствами 1С.

Очевидно, что разработка «подготовительного» программного обеспечения может выполняться заблаговременно. Причем если предусмотреть сохранение результатов в некоторый стандартизованный формат, по своей структуре максимально приближенный к требуемому в 1С, становятся возможными неоднократная «виртуальная» загрузка данных и их промежуточная верификация, в том числе на основе сравнения с исходными данными. За счет этого количество ошибок передачи исходной информации может быть сведено к нулю.

Далее в контексте настоящего документа предполагается наличие описанного выше «подготовительного» программного обеспечения и сохранение результатов его работы в промежуточных файлах или реляционной базе данных, логическая[[1]](#footnote-1) структура которых совпадает с требуемой в 1С.

# Обзор технологий загрузки данных в 1С

Загрузка данных в 1С является частным случаем ее интеграции [[[2]](#endnote-1)] с внешними программами и может выполняться следующими способами:

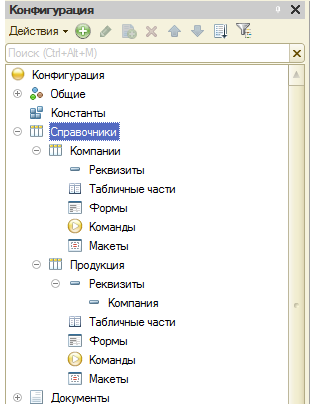
1. [Импорт внешних файлов (XML, Excel, DBF, текстовые файлы)](#_Импорт_внешних_файлов).
2. [Передача данных через WEB-сервисы 1С](#_Передача_данных_через).
3. [Использование встроенного в 1С механизма обмена данными (планы обмена)](#_Использование_встроенного_механизма).
4. [Использование встроенного в 1С механизма доступа к внешним ODBC-данным](#_Использование_встроенного_механизма_1).
5. [Вызов в 1С-скрипте пользовательской внешней компоненти и чтение данных через ее методы (свойства)](#_Вызов_в_1С-скрипте).
6. [Вызов в 1С-скрипте пользовательского COM-объекта и чтение данных через его методы (свойства)](#_Вызов_пользовательского_COM-объекта).
7. [Вызов в 1С-скрипте COM-объекта ADO, обеспечеивающего доступ к внешним OLEDB-данным](#_Вызов_COM-объекта_ADO).

1. [Вызов 1С в качестве COM-сервера](#_Вызов_1С_в).

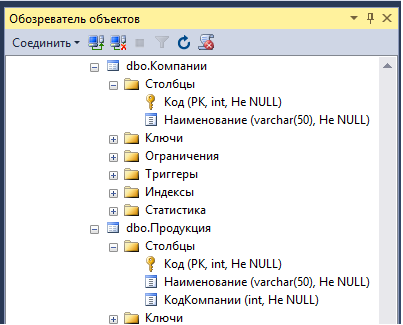
В случае развертывания 1С на базе данных SQL-типа (Microsoft SQL Server, Oracle, IBM DB2 или PostgreSQL) как частный случай может использоваться [прямая загрузка в ее базу данных](#_Прямая_загрузка_в).[[3]](#footnote-2)

Далее перечисленные способы будут рассмотрены более подробно, при этом в качестве основных критериев оценки будут рассматриваться быстродействие и простота кода как на стороне 1С, так и в приложении загрузки (при его наличии).

Для упрощения кода загрузки данные будут загружаться в два логически связанных справочника 1С – Компании и Продукция:



В качестве исходных данных (если не оговорено иное) будут использоваться следующие таблицы на Microsoft SQL Server, эквивалентные соответствующим справочникам 1С:



Для иллюстрации кода приложения загрузки (при наличии такого кода и если не оговорено иное) будет использоваться C#.

## Импорт внешних файлов

Общий подход к загрузке данных способом импорта внешних файлов заключается в том, что «подготовительная» программа (см. выше) записывает результаты своей работы в виде одного или нескольких файлов, а 1С импортирует эти файлы, интерпретирует их содержимое и добавляет данные в соответствующие справочники.

Наиболее распространенные и простые для реализации в «подготовительной» программе форматы – текстовые файлы и XML.

### Импорт текстовых файлов

#### Краткое описание технологии

Самый простой способ представить табличные данные в текстовом файле – использовать строки с разделителями, в качестве которых используется какой-либо символ, например, "|":

1. Компании.txt (Код, Наименование):

1|Компания 1

2|Компания 2

…

1. Продукция.txt (Код, Наименование, КодКомпании):

1|Продукция 1-1|1

2|Продукция 1-2|1

3|Продукция 2-1|2

…

Для небольших по объему файлов можно использовать чтение всего содержимого в память, для больших – построчное (критический объем файлов определяется экспериментальным путем).

#### Псевдокод

1С

читать файл Компании, для каждой строки:

разобрать строку на значения полей;

формировать новую запись справочника Компании;

добавить новую запись в справочник Компании;

читать файл Продукция, для каждой строки:

разобрать строку на значения полей, определить КодКомпании;

РодительскаяЗаписьКомпании = НайтиЗаписьКомпании(КодКомпании);

формировать новую запись справочника Продукция с учетом РодительскаяЗаписьКомпании;

добавить новую запись в справочник Продукция;

#### Оптимизация

В представленном псевдокоде выделен поиск родительской записи в справочнике Компании, который будет выполняться для каждой записи справочника Продукция.

Арность отношения этих справочников 1:N, соответственно при N = 1..X ожидаемое соотношение количества записей в указанных справочниках составит 1:1.5\*(X-1). Практическое следствие – поиск одних и тех же записей в справочнике Компании будет выполняться неоднократно.

Количество операций такого рода можно уменьшить до MIN(записейКомпании, записейПродукция) – достаточно обеспечить предварительную сортировку записей исходного справочника Продукция по коду компании (псевдокод):

ПоследнийКодКомпании = 0;

РодительскаяЗаписьКомпании;

читать файл Продукция, для каждой строки:

разобрать строку на значения полей, определить КодКомпании;

если КодКомпании <> ПоследнийКодКомпании

{

ПоследнийКодКомпании = КодКомпании;

РодительскаяЗаписьКомпании = НайтиЗаписьКомпании(ПоследнийКодКомпании);

}

формировать новую запись справочника Продукция с учетом РодительскаяЗаписьКомпании;

добавить новую запись в справочник Продукция;

Очевидный минус такой оптимизации – рост алгоритмической сложности кода 1С с увеличением количества логических связей между справочниками.

#### Пример кода 1

1С – чтение всего содержимого текстового файла в память

Текст = Новый ТекстовыйДокумент;

Текст.Прочитать("Компании.txt", КодировкаТекста.ANSI, Символы.ПС);

Для Номер = 1 По Текст.КоличествоСтрок() Цикл

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

МассивСтрок = Утилиты.РазложитьСтрокуВМассивПодстрок(Текст.ПолучитьСтроку(Номер), "|");

НоваяКомпания.Код = МассивСтрок[0];

НоваяКомпания.Наименование = МассивСтрок[1];

НоваяКомпания.Записать();

КонецЦикла;

Текст.Прочитать("Продукция.txt", КодировкаТекста.ANSI, Символы.ПС);

ПоследнийКодКомпании = 0;

РодительскаяЗаписьКомпании = Неопределено;

Для Номер = 1 По Текст.КоличествоСтрок() Цикл

МассивСтрок = Утилиты.РазложитьСтрокуВМассивПодстрок(Текст.ПолучитьСтроку(Номер), "|");

КодКомпании = МассивСтрок[2];

Если КодКомпании <> ПоследнийКодКомпании Тогда

ПоследнийКодКомпании = КодКомпании;

РодительскаяЗаписьКомпании = Справочники.Компании.НайтиПоКоду(ПоследнийКодКомпании);

КонецЕсли;

Если РодительскаяЗаписьКомпании <> Неопределено Тогда

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = МассивСтрок[0];

НоваяПродукция.Наименование = МассивСтрок[1];

НоваяПродукция.Компания = РодительскаяЗаписьКомпании.Ссылка;

НоваяПродукция.Записать();

КонецЕсли;

КонецЦикла;

##### Особенности кода

1. Используется пользовательская функция РазложитьСтрокуВМассивПодстрок (общий модуль Утилиты), которая получает строку для разбора и разделитель, а возвращает массив строк, соответствующих подстрокам.

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 57 сек

##### Недостатки

1. Так как все данные загружаются в память, могут обрабатываться относительно небольшие объемы информации.

#### Пример кода 2

1С – построчное чтение текстового файла

Файл = Новый ЧтениеТекста("Компании.txt", КодировкаТекста.ANSI);

Пока Истина Цикл

Строка = Файл.ПрочитатьСтроку();

Если Строка <> Неопределено Тогда

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

МассивСтрок = Утилиты.РазложитьСтрокуВМассивПодстрок(Строка, "|");

НоваяКомпания.Код = МассивСтрок[0];

НоваяКомпания.Наименование = МассивСтрок[1];

НоваяКомпания.Записать();

Иначе

Прервать;

КонецЕсли;

КонецЦикла;

Файл.Закрыть();

Файл = Новый ЧтениеТекста("Продукция.txt", КодировкаТекста.ANSI);

ПоследнийКодКомпании = 0;

РодительскаяЗаписьКомпании = Неопределено;

Пока Истина Цикл

Строка = Файл.ПрочитатьСтроку();

Если Строка <> Неопределено Тогда

МассивСтрок = Утилиты.РазложитьСтрокуВМассивПодстрок(Строка, "|");

КодКомпании = МассивСтрок[2];

Если КодКомпании <> ПоследнийКодКомпании Тогда

ПоследнийКодКомпании = КодКомпании;

РодительскаяЗаписьКомпании = Справочники.Компании.НайтиПоКоду(ПоследнийКодКомпании);

КонецЕсли;

Если РодительскаяЗаписьКомпании <> Неопределено Тогда

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = МассивСтрок[0];

НоваяПродукция.Наименование = МассивСтрок[1];

НоваяПродукция.Компания = РодительскаяЗаписьКомпании.Ссылка;

НоваяПродукция.Записать();

КонецЕсли;

Иначе

Прервать;

КонецЕсли;

КонецЦикла;

Файл.Закрыть();

##### Особенности кода

1. Используется пользовательская функция РазложитьСтрокуВМассивПодстрок (общий модуль Утилиты), которая получает строку для разбора и разделитель, а возвращает массив строк, соответствующих подстрокам.

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 58 сек

##### Преимущества

1. Могут загружаться файлы большого объема.

### Импорт XML-файлов

#### Краткое описание технологии

XML-файлы могут формироваться аналогично текстовым (см. выше), но при этом отражать иерархическую зависимость записей, например:

<?xml version="1.0"?>

<Данные>

<Компания код="1" наименование="Компания 1">

<Продукция код="1" наименование="Продукция 1-1"/>

<Продукция код="2" наименование="Продукция 1-2"/>

</Компания>

<Компания код="2" наименование="Компания 2">

<Продукция код="3" наименование="Продукция 2-1"/>

</Компания>

</Данные>

Как и для текстовых файлов, для небольших по объему XML-файлов можно использовать чтение всего содержимого в память, для больших – последовательное (критический объем файлов определяется экспериментальным путем).

В случае чтения всего содержимого XML-файла в память может использоваться прямой доступ к тэгам и атрибутам XML или десериализация 1С-объектов с помощью XDTO. Пакеты XDTO могут применяться и в гибридной схеме чтения XML, когда XML-файл читается последовательно, а его отдельные фрагменты (теги) десериализуются с помощью XDTO.

Пакет XDTO, соответствующий XML-файлу исходных данных, получен импортом в 1С XSD-схемы исходного файла:

<xs:schema

xmlns:tns="http://localhost/xdto"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://localhost/xdto"

attributeFormDefault="unqualified"

elementFormDefault="qualified">

<xs:element name="Данные">

<xs:complexType>

<xs:sequence>

<xs:element name="Компания" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="Продукция" maxOccurs="unbounded">

<xs:complexType>

<xs:attribute name="код"

type="xs:unsignedInt" use="required"/>

<xs:attribute name="наименование"

type="xs:string" use="required"/>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute name="код" type="xs:unsignedInt" use="required"/>

<xs:attribute name="наименование" type="xs:string" use="required"/>

</xs:complexType>

</xs:element>

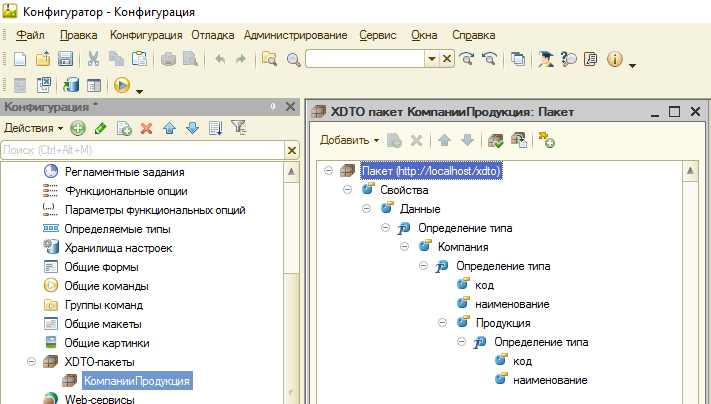
</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Получившийся пакет XDTO:



Варианты загрузки XML:

|  |  |  |
| --- | --- | --- |
|  | Чтение всего XML в память | Последовательное чтение |
| Прямой доступ к тэгам и атрибутам XML | [Пример кода 1](#_Пример_кода_1) | [Пример кода 2](#_Пример_кода_2) |
| Десериализация с использованием XDTO | [Пример кода 3](#_Пример_кода_3) | [Пример кода 4](#_Пример_кода_4) |

#### Пример кода 1

1С – чтение всего содержимого XML-файла в память, прямой доступ к тэгам и атрибутам

ЧтениеXML = Новый ЧтениеXML;

ЧтениеXML.ОткрытьФайл("Данные.xml");

ПостроительDOM = Новый ПостроительDOM;

ДокументDOM = ПостроительDOM.Прочитать(ЧтениеXML);

ЧтениеXML.Закрыть();

Для Каждого xmlКомпания Из ДокументDOM.ЭлементДокумента.ДочерниеУзлы Цикл

Если НЕ (xmlКомпания.ТипУзла = ТипУзлаDOM.Элемент И xmlКомпания.ИмяУзла = "Компания") Тогда

Продолжить;

КонецЕсли;

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

НоваяКомпания.Код =

xmlКомпания.Атрибуты.ПолучитьИменованныйЭлемент("код").ЗначениеУзла;

НоваяКомпания.Наименование =

xmlКомпания.Атрибуты.ПолучитьИменованныйЭлемент("наименование").ЗначениеУзла;

НоваяКомпания.Записать();

Для Каждого xmlПродукция из xmlКомпания.ДочерниеУзлы Цикл

Если НЕ (xmlПродукция.ТипУзла = ТипУзлаDOM.Элемент

И xmlПродукция.ИмяУзла = "Продукция") Тогда

Продолжить;

КонецЕсли;

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код =

xmlПродукция.Атрибуты.ПолучитьИменованныйЭлемент("код").ЗначениеУзла;

НоваяПродукция.Наименование =

xmlПродукция.Атрибуты.ПолучитьИменованныйЭлемент("наименование").ЗначениеУзла;

НоваяПродукция.Компания = НоваяКомпания.Ссылка;

НоваяПродукция.Записать();

КонецЦикла;

КонецЦикла;

##### Особенности кода

1. XML естественным образом отражает иерархическую связь между компаниями и их продукцией, поэтому формирование новых записей соответствующих справочников осуществляется в соответствии с этой иерархией и поиск родительских записей для новых записей справочника Продукция не требуется.

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 59 сек

##### Недостатки

1. Так как все данные загружаются в память, могут обрабатываться относительно небольшие объемы информации.
2. Требуется большой объем вспомогательного кода для доступа к значениям тэгов и атрибутов XML.
3. Низкая наглядность кода.

#### Пример кода 2

1С – последовательное чтение содержимого XML-файла, прямой доступ к тэгам и атрибутам

ЧтениеXML = Новый ЧтениеXML;

ЧтениеXML.ОткрытьФайл("Данные.xml");

НоваяКомпания = Неопределено;

Пока ЧтениеXML.Прочитать() Цикл

Если ЧтениеXML.ТипУзла = ТипУзлаXML.НачалоЭлемента Тогда

Если ЧтениеXML.Имя = "Компания" Тогда

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

Пока ЧтениеXML.ПрочитатьАтрибут() Цикл

Если ЧтениеXML.Имя = "код" Тогда

НоваяКомпания.Код = ЧтениеXML.Значение;

ИначеЕсли ЧтениеXML.Имя = "наименование" Тогда

НоваяКомпания.Наименование = ЧтениеXML.Значение;

КонецЕсли;

КонецЦикла;

НоваяКомпания.Записать();

ИначеЕсли ЧтениеXML.Имя = "Продукция" И НоваяКомпания <> Неопределено Тогда

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

Пока ЧтениеXML.ПрочитатьАтрибут() Цикл

Если ЧтениеXML.Имя = "код" Тогда

НоваяПродукция.Код = ЧтениеXML.Значение;

ИначеЕсли ЧтениеXML.Имя = "наименование" Тогда

НоваяПродукция.Наименование = ЧтениеXML.Значение;

КонецЕсли;

КонецЦикла;

НоваяПродукция.Компания = НоваяКомпания.Ссылка;

НоваяПродукция.Записать();

КонецЕсли;

КонецЕсли;

КонецЦикла;

ЧтениеXML.Закрыть();

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 59 сек

#### Пример кода 3

1С – чтение всего содержимого XML-файла в память, десериализация 1С-объектов с помощью XDTO

ЧтениеXML = Новый ЧтениеXML;

ЧтениеXML.ОткрытьФайл("C:\Проекты\1S\DataGeneratorXML\bin\Debug\Data.xml");

ТипКомпанииПродукция = ФабрикаXDTO.Тип("http://localhost/xdto", "Данные");

КомпанииПродукцияXDTO = ФабрикаXDTO.ПрочитатьXML(ЧтениеXML, ТипКомпанииПродукция);

ПоследовательностьКомпании = КомпанииПродукцияXDTO.Последовательность();

КоличествоКомпании = ПоследовательностьКомпании.Количество();

Для I = 0 По КоличествоКомпании - 1 Цикл

КомпанияXDTO = ПоследовательностьКомпании.ПолучитьЗначение(I);

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

НоваяКомпания.Код = КомпанияXDTO.код;

НоваяКомпания.Наименование = КомпанияXDTO.наименование;

НоваяКомпания.Записать();

НоваяКомпанияСсылка = НоваяКомпания.Ссылка;

ПоследовательностьПродукция = КомпанияXDTO.Последовательность();

КоличествоПродукция = ПоследовательностьПродукция.Количество();

Для J = 0 По КоличествоПродукция - 1 Цикл

ПродукцияXDTO = ПоследовательностьПродукция.ПолучитьЗначение(J);

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = ПродукцияXDTO.код;

НоваяПродукция.Наименование = ПродукцияXDTO.наименование;

НоваяПродукция.Компания = НоваяКомпанияСсылка;

НоваяПродукция.Записать();

КонецЦикла;

КонецЦикла;

##### Особенности кода

1. При использовании XDTO 1С автоматически десериализует значения элементов и атрибутов исходного XML в соответствии с заданной схемой (см. выделенные фрагменте в коде).

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 79 сек

#### Пример кода 4

Последовательное чтение содержимого XML-файла в сочетании с десериализацией 1С-объектов с помощью XDTO (см. <http://infostart.ru/public/418363/> ) не проверялось, т.к. существенной разницы между временем обработки XML-файла, загружаемого в память, и построчного его чтения не выявлено. Использование же десериализации с помощью XDTO приводит к полутарократному замедлению – примерно таких же результатов можно ожидать и при потоковом чтении XML-файла.

#### Особенности представления данных в XML

XML представляет собой иерархию узлов, что хорошо подходит для простого примера. Однако необходимо понимать, что на практике реальные исходные данные представить в виде одной иерархии скорее всего будет сложно, если вообще возможно. В качестве иллюстрации этого утверждения рассмотрим простую модель данных исходной информации о персонале, построенной с использованием Общероссийского классификатора информации о населении (ОКИН):

Таблица Работники

* ФИО
* Пол – код по ОКИН
* Гражданство – код по ОКИН
* …

Отмеченные маркером поля содержат коды, соответствующие определенным записям в физических или виртуальных справочниках однотипной структуры:

Таблица СправочникПол

* Код
* Наименование

В приведенной модели присутствует две зависимости вида «Работник -> Справочник», что не позволяет спроецировать исходные данные в естественную иерархию XML.

Решается эта проблема или использованием ссылок XML (для небольших файлов, загружаемых в память), или представлением в XML набора «плоских» данных (в одном или нескольких файлах). При этом преимущество XML как иерархического представления данных исчезает и логика обработки исходной информации становится аналогичной [загрузке данных из текстовых файлов](#_Импорт_текстовых_файлов), включая необходимость поиска родительских записей в справочниках.

##### Пример ссылок XML

<?xml version="1.0"?>

<Данные>

<Работники>

<Работник фио="Иванов И.И.">

<ref:Пол ref="01-1"/>

<ref:Гражданство ref="02-1"/>

</Работник>

<Работник фио="Петрова Н.П.">

<ref:Пол ref="01-2"/>

<ref:Гражданство ref="02-1"/>

</Работник>

</Работники>

<СправочникПол>

<data:Пол id="01-1" Наименование="Мужской" />

<data:Пол id="01-2" Наименование="Женский" />

</СправочникПол>

<СправочникГражданство>

<data:Гражданство id="02-1" Наименование="Гражданин Российской Федерации" />

<data:Гражданство id="02-2" Наименование="Гражданин Российской Федерации и иностранного государства (двойное гражданство)" />

<data:Гражданство id="02-3" Наименование="Иностранный гражданин" />

<data:Гражданство id="02-4" Наименование="Лицо без гражданства" />

</СправочникГражданство>

</Данные>

##### Пример XML с набором «плоских» данных

<?xml version="1.0"?>

<Данные>

<Работники>

<Работник фио="Иванов И.И." кодПол="01-1" кодГражданство="02-1"/>

<Работник фио="Петрова Н.П." кодПол="01-2" кодГражданство="02-1"/>

</Работники>

<СправочникПол>

<Пол код="01-1" Наименование="Мужской" />

<Пол код="01-2" Наименование="Женский" />

</СправочникПол>

<СправочникГражданство>

<Гражданство код="02-1" Наименование="Гражданин Российской Федерации" />

<Гражданство код="02-2" Наименование="Гражданин Российской Федерации и иностранного государства (двойное гражданство)" />

<Гражданство код="02-3" Наименование="Иностранный гражданин" />

<Гражданство код="02-4" Наименование="Лицо без гражданства" />

</СправочникГражданство>

</Данные>

### Импорт JSON-файлов[[4]](#footnote-3)

#### Краткое описание технологии

Формат JSON так же, как и XML, позволяет легко компоновать иерархические данные, но в отличие от XML является более лаконичным, что делает его более экономным при передаче данных.

Пример JSON-файла, эквивалентного XML в разделе [Импорт XML-файлов](#_Импорт_XML-файлов):

[

{

"код": "1",

"наименование": "Компания 1",

"Продукция": [

{

"код": "1",

"наименование": "Продукция 1-1"

},

{

"код": "2",

"наименование": "Продукция 1-2"

}]

},

{

"код": "2",

"наименование": "Компания 2",

"Продукция": [

{

"код": "3",

"наименование": "Продукция 2-1"

}]

}

]

Так же, как и для XML, в 1С предусмотрены как десериализация содержимого JSON-файла в памяти, так и его потоковое чтение.

#### Пример кода 1

1С – десериализация содержимого JSON-файла в памяти

ЧтениеJSON = Новый ЧтениеJSON();

ЧтениеJSON.ОткрытьФайл("C:\Проекты\1S\DataGeneratorJSON\bin\Debug\Data.json");

КомпанииJSON = ПрочитатьJSON(ЧтениеJSON);

ЧтениеJSON.Закрыть();

Для Каждого КомпанияJSON Из КомпанииJSON Цикл

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

НоваяКомпания.Код = КомпанияJSON.код;

НоваяКомпания.Наименование = КомпанияJSON.наименование;

НоваяКомпания.Записать();

НоваяКомпанияСсылка = НоваяКомпания.Ссылка;

Для Каждого ПродукцияJSON Из КомпанияJSON.Продукция Цикл

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = ПродукцияJSON.код;

НоваяПродукция.Наименование = ПродукцияJSON.наименование;

НоваяПродукция.Компания = НоваяКомпанияСсылка;

НоваяПродукция.Записать();

КонецЦикла;

КонецЦикла;

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 63 сек

#### Пример кода 2

Потоковое чтение JSON-файла требует слишком сложного кода (см. <http://v8.1c.ru/o7/201410json/> ), а поскольку преимущества использования JSON для представления исходных данных в сравнении с [текстовыми файлами](#_Импорт_текстовых_файлов) и [XML](#_Импорт_XML-файлов) неочевидны, тестироваться этот режим не будет.

## Передача данных через WEB-сервисы 1С

#### Краткое описание технологии

Использование WEB-сервисов для интеграции внешнего приложения с 1С оправдано только в случаях, когда 1С и/или внешнее приложение работает не на Windows-платформе либо требуется обеспечить доступ к 1С через интернет (в зависимости от необходимой функциональности вместо WEB-сервисов могут использоваться альтернативные решения, например, тонкий клиент 1С или 1С:Битрикс).

Кроме того, WEB-сервисы плохо приспособлены для передачи больших объемов данных и работают в сравнении с другими технологиями существенно медленнее. С учетом этого, а также из-за необходимости разработки специального клиентского приложения рассматривать WEB-сервисы как потенциальную технологию для загрузки данных нецелесообразно.

Примечание: Причина сравнительно медленной работы WEB-сервисов заключается в их внутреннем механизме, который можно проиллюстрировать на примере типового запроса от клиентского приложения к WEB-сервису:

1. клиент (в памяти): сериализует в XML, определяемый WSDL-схемой сервиса, все параметры, которые должны передаваться методу WEB-сервиса;
2. клиент: формирует GET или PUT запрос (определяется WSDL-схемой сервиса), в тело которого включается сформированный на предыдущем этапе XML, и отправляет его через http-протокол на url-адрес сервиса;
3. транспортный уровень: разбивает http-запрос на отдельные пакеты, передает их с подтверждением доставки;
4. WEB-сервер: получает http-запрос, проверяет его корректность и аутентификацию, активизирует WEB-сервис и перенаправляет ему тело запроса;
5. WEB-сервис (в памяти): десериализует полученный XML во внутреннее (объектное) представление и выполняет запрошенный метод;
6. WEB-сервис (в памяти): сериализует в XML результаты работы;
7. WEB-сервис: формирует http-ответ, в тело которого включается сформированный на предыдущем этапе XML, и отправляет его через http-протокол на url-адрес клиента;
8. транспортный уровень: разбивает http-ответ на отдельные пакеты, передает их с подтверждением доставки;
9. клиент: получает http-ответ, проверяет его корректность и идентификацию;
10. клиент (в памяти): десериализует полученный в теле http-ответа XML во внутреннее (объектное) представление и выполняет обработку полученных результатов.

Очевидно, что сериализация в XML и десериализация из XML (1, 5, 6, 10) требуют значительных ресурсов – как памяти, так и процессорного времени.

## Использование встроенного механизма обмена данными

#### Краткое описание технологии

Как один из механизмов интеграции с другими программами (данными) в 1С предусмотрен «Универсальный механизм обмена данными» [[[5]](#endnote-2)], на основе которого строятся, например, территориально распределенные системы на основе 1С:Предприятия 8.

По сути этот механизм основан на обмене XML-сообщениями, а с учетом его дополнительных функций типа Службы регистрации изменений логично предполагать, что время загрузки при использовании «Универсального механизма обмена данными» будет больше, чем непосредственное применение XML.

На основании этого принимается решение не тестировать указанный механизм загрузки данных.

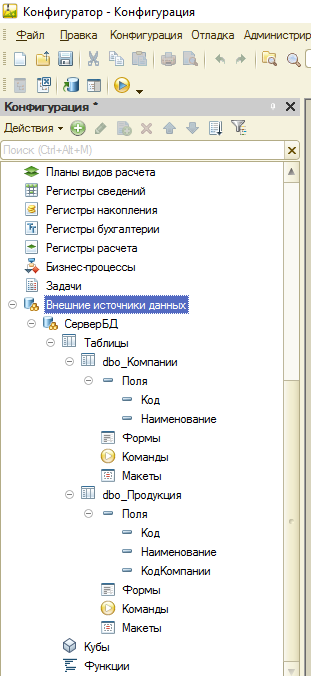
## Использование встроенного механизма доступа к внешним ODBC-данным[[6]](#footnote-4)

#### Краткое описание технологии

В 1С реализована встроенная возможность получать данные из внешних источников с использованием интерфейса ODBC. Для доступа к данным внешних источников используется объектная модель 1С и язык запросов.

Порядок подключения внешних источников данных описан, например, на сайте [[[7]](#endnote-3)].

Подключенные внешние источники:



#### Пример кода

1С

Соединение = Новый ПараметрыСоединенияВнешнегоИсточникаДанных;

Соединение.СтрокаСоединения = "DRIVER={SQL Server};SERVER=S\_Server;UID=User;PWD=Password;

|DATABASE=S\_DATABASE;LANGUAGE=русский";

Соединение.ИмяПользователя = "User";

Соединение.Пароль = "Password";

ВнешниеИсточникиДанных.СерверБД.УстановитьОбщиеПараметрыСоединения(Соединение);

ВнешниеИсточникиДанных.СерверБД.УстановитьСоединение();

Запрос = Новый Запрос(

"ВЫБРАТЬ dbo\_Компании.Код, dbo\_Компании.Наименование

|ИЗ ВнешнийИсточникДанных.СерверБД.Таблица. dbo\_Компании КАК dbo\_Компании");

Выборка = Запрос.Выполнить().Выбрать();

Пока Выборка.Следующий() Цикл

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

НоваяКомпания.Код = Выборка.Код;

НоваяКомпания.Наименование = Выборка.Наименование;

НоваяКомпания.Записать();

КонецЦикла;

ПоследнийКодКомпании = 0;

РодительскаяЗаписьКомпании = Неопределено;

Запрос = Новый Запрос(

"ВЫБРАТЬ dbo\_Продукция.Код, dbo\_Продукция.Наименование, dbo\_Продукция.КодКомпании

|ИЗ ВнешнийИсточникДанных.СерверБД.Таблица. dbo\_Продукция КАК dbo\_Продукция

|УПОРЯДОЧИТЬ ПО dbo\_Продукция.КодКомпании");

Выборка = Запрос.Выполнить().Выбрать();

Пока Выборка.Следующий() Цикл

Если Выборка.КодКомпании <> ПоследнийКодКомпании Тогда

ПоследнийКодКомпании = Выборка.КодКомпании;

РодительскаяЗаписьКомпании = Справочники.Компании.НайтиПоКоду(ПоследнийКодКомпании);

КонецЕсли;

Если РодительскаяЗаписьКомпании <> Неопределено Тогда

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = Выборка.Код;

НоваяПродукция.Наименование = Выборка.Наименование;

НоваяПродукция.Компания = РодительскаяЗаписьКомпании.Ссылка;

НоваяПродукция.Записать();

КонецЕсли;

КонецЦикла;

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 89 сек

## Вызов в 1С-скрипте пользовательской внешней компоненты

#### Краткое описание технологии

Технология внешних компонент в 1С [[[8]](#endnote-4)] позволяет создавать программы, которые будут динамически подключаться и тесно взаимодействовать с системой «1С:Предприятие», расширяя ее возможности.

Технически внешняя компонента представляют собой динамически подключаемую библиотеку, для которой стандартизован набор экспортируемых функций.

Для Windows это dll-библиотека с фактическим внутрипроцессорным COM-сервером. И хотя такая библиотека не требует регистрации в реестре Windows, каких-либо преимуществ у нее в сравнении с традиционным COM нет (для вызова экспортируемых функций даже используются COM-совместимые типы данных).

Очевидно, что разработка внешней компоненты с использованием Native API для C гораздо сложнее создания COM-объекта на VB, Delphi или C#. А поскольку регистрация COM-объектов при загрузке данных проблем не представляет, важнее оценить другие преимущества и недостатки технологии внешних компонент 1С в сравнении с COM-объектами.

Теоретическое преимущество технологии внешних компонент – возможность использования в пользовательском коде менеджера памяти 1С, что позволяет скрипту 1С корректно удалять ненужные объекты внешней компоненты. Однако необязательность его использования сводит это преимущество на нет. Более того, в COM существует встроенный механизм на основе подсчета ссылок, позволяющий корректно и автоматически удалять ненужные объекты.

Других преимуществ не просматривается.

Недостатки технологии внешних компонент:

1. внешние компоненты могут «отдавать» только простые типы данных и даже с массивами работа сильно усложняется как в C-коде, так и в 1С;
2. вложенные объекты не поддерживаются;
3. большой объем вспомогательного C-кода, обеспечивающего согласование типов данных, а также имитацию рефлексии и поддержку мультиязычности в 1С.

С учетом сложности кода рассмотренных примеров реализации внешних компонент и ввиду отсутствия преимуществ в сравнении с COM-объектами принято решение внешние компоненты 1С для загрузки данных не использовать.

## Использование в 1С-скрипте COM-объектов

### Вызов пользовательского COM-объекта

#### Краткое описание технологии

Поскольку язык 1С поддерживает работу с внешними COM-объектами, можно написать собственный COM-объект, который будет предоставлять 1С доступ к внешним данным.

Диапазон возможностей для разработчика ничем неограничен – можно получать и возвращать не только простые типы данных, но также массивы и составные объекты, можно использовать COM-итераторы [[[9]](#endnote-5)], и т.д.

Ниже представлен пример, где для перебора компаний использован COM-итератор, записи продукции представлены простым массивом, а каждая запись компании представляют собой массив, один из элементов которого является итератором по записям выпускаемой продукции.

#### Пример кода

C#

[Guid("AC2275BD-D8E9-4EC5-988C-3BCE4E1E92BE")]

public interface IDataLoader

{

DataEnumerable GetData();

}

[Guid("0D569C21-0DD4-4E14-8338-5B7A70CB0433"), ClassInterface(ClassInterfaceType.None)]

public class DataLoader : IDataLoader

{

public DataEnumerable GetData()

{

return new DataEnumerable(

исходныеДанные.ПолучитьКомпании()

.Select(company => new object[]

{

company.Код,

company.Наименование,

new DataEnumerable(

исходныеДанные.ПолучитьПродукцию(company)

.Select(product => new object[]

{

product.Код,

product.Наименование,

company.Код

}))

}));

}

}

1С

МойОбъект = Новый COMОбъект("Com1For1C.DataLoader");

Компании = МойОбъект.GetData();

Для Каждого КомпанияДанные из Компании Цикл

Компания = Справочники.Компании.СоздатьЭлемент();

Компания.Код = КомпанияДанные.GetValue(0);

Компания.Наименование = КомпанияДанные.GetValue(1);

Компания.Записать();

КомпанияСсылка = Компания.Ссылка;

Для Каждого ПродукцияДанные из КомпанияДанные.GetValue(2) Цикл

Продукция = Справочники.Продукция.СоздатьЭлемент();

Продукция.Код = ПродукцияДанные.GetValue(0);

Продукция.Наименование = ПродукцияДанные.GetValue(1);

Продукция.Компания = КомпанияСсылка;

Продукция.Записать();

КонецЦикла;

КонецЦикла;

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 61 сек

##### Недостатки

1. Требуется кодирование как в 1С, так и в C#.

### Вызов COM-объекта ADO

#### Краткое описание технологии

Очевидно, что написание пользовательских COM-объектов требует дополнительных трудозатрат. Если же исходные данные хранятся в некоторой внешней базе или другом источнике данных, доступном через ODBC, OLE DB или проприетарные драйверы, совместимые с ADO, для чтения информации из внешней базы данных можно использовать инфраструктуру ADO [[[10]](#endnote-6)].

#### Пример кода

1С

СтрокаСоединения =

"Provider=SQLNCLI11;Server=.\sqlexpress;Database=Test1S;Uid=myUsername;Pwd=myPassword;";

con = Новый COMОбъект("ADODB.Connection");

con.Open(СтрокаСоединения);

cmd = Новый COMОбъект("ADODB.Command");

cmd.ActiveConnection = con;

cmd.CommandType = 1; // adCmdText

rs = Новый ComОбъект("ADODB.RecordSet");

rs.CursorType = 0; // adOpenForwardOnly

rs.LockType = 1; // adLockReadOnly

cmd.CommandText = "select \* from Компании";

rs = cmd.Execute();

Пока rs.EOF() = 0 Цикл

НоваяКомпания = Справочники.Компании.СоздатьЭлемент();

НоваяКомпания.Код = rs.Fields("Код").Value;

НоваяКомпания.Наименование = rs.Fields("Наименование").Value;

НоваяКомпания.Записать();

rs.MoveNext();

КонецЦикла;

rs.Close();

ПоследнийКодКомпании = 0;

РодительскаяЗаписьКомпании = Неопределено;

cmd.CommandText = "select \* from Продукция order by КодКомпании";

rs = cmd.Execute();

Пока rs.EOF() = 0 Цикл

Если rs.Fields("КодКомпании").Value <> ПоследнийКодКомпании Тогда

ПоследнийКодКомпании = rs.Fields("КодКомпании").Value;

РодительскаяЗаписьКомпании = Справочники.Компании.НайтиПоКоду(ПоследнийКодКомпании);

КонецЕсли;

Если РодительскаяЗаписьКомпании <> Неопределено Тогда

НоваяПродукция = Справочники.Продукция.СоздатьЭлемент();

НоваяПродукция.Код = rs.Fields("Код").Value;

НоваяПродукция.Наименование = rs.Fields("Наименование").Value;

НоваяПродукция.Компания = РодительскаяЗаписьКомпании.Ссылка;

НоваяПродукция.Записать();

КонецЕсли;

rs.MoveNext();

КонецЦикла;

rs.Close();

con.Close();

##### Особенности кода:

1. Подключаемая база данных определяется строкой соединения [[[11]](#endnote-7)] – переменная СтрокаСоединения.
2. Поскольку ADO не поддерживает режим множественных активных наборов данных MARS [[[12]](#endnote-8)], информация из исходных таблиц читается поочередно.
3. При формировании записей «дочернего» справочника Продукция необходимо найти «родительскую» запись справочника Компании и установить на нее ссылку. Для повышения быстродействия этих операций записи исходной таблицы Продукция сортируются по коду компании-владельца, а алгоритм добавления новый записей в справочник Продукция оптимизирован в сторону уменьшения операций поиска в справочнике Компании аналогично [оптимизации загрузки текстовых файлов](#_Оптимизация) и [чтению данных из внешних источников](#_Использование_встроенного_механизма_1).

##### Оценка быстродействия

50 компаний х 100 видов продукции = 50 + 5000 = 5050 записей -> 63 сек

##### Недостатки

1. Сложность 1С-кода, обусловленная смешением логики обработки исходных записей и особенностей управления соединениями, командами и наборами данных.

## Вызов 1С в качестве COM-сервера

#### Краткое описание технологии

1С поддерживает не только стандарт интеграции программ OLE Automation, но и более эффективную возможность использования 1С во внешней программе в виде inproc-сервера COM. Этот механизм в 1С называется «Внешнее соединение» и позволяет внешнему приложению обращаться к предоставляемым через этот программный интерфейс методам 1С.

Ниже приведен пример кода добавления записей в файловую базу данных 1С.

#### Пример кода

C#

COMConnector comConnector = null;

dynamic v8\_base, newCompany, newProduct;

try

{

comConnector = new COMConnector();

comConnector.MaxConnections = 2;

string каталогБД = ConfigurationManager.AppSettings["каталогБД"];

v8\_base = comConnector.Connect(string.Format("File=\"{0}\"", каталогБД));

foreach (var company in компанииПродукция)

{

var компания = company.Key;

newCompany = v8\_base.Справочники.Компании.СоздатьЭлемент();

newCompany.Код = компания.Код;

newCompany.Наименование = компания.Наименование;

newCompany.Записать();

object ссылка = newCompany.Ссылка;

foreach (var product in company)

{

newProduct = v8\_base.Справочники.Продукция.СоздатьЭлемент();

newProduct.Код = product.Код;

newProduct.Наименование = product.Наименование;

newProduct.Компания = ссылка;

newProduct.Записать();

}

}

}

finally

{

newProduct = null;

newCompany = null;

v8\_base = null;

if (comConnector != null)

{

Marshal.FinalReleaseComObject(comConnector);

comConnector = null;

GC.Collect();

}

}

##### Оценка быстродействия

Поскольку в коде C# происходит многократное обращение к COM-объектам 1С, загрузка выполняется чрезвычайно медленно – 110 записей загружались 24 сек.

## Прямая загрузка в базу данных 1С

### Общее решение

Все рассмотренные выше способы загрузки объединяет необходимость выполнения на стороне 1С определенного кода, который:

1. читает исходные данные через «канал связи» и интерпретирует их;
2. формирует новые записи соответствующих справочников;
3. выполняет поиск родительских записей для обеспечения логической целостности данных;
4. сохраняет информацию в базе данных 1С.

Замеры времени выполнения отдельных фаз загрузки для разных способов показали, что первая фаза, если не используются «тяжеловесные» механизмы десериализации XML и JSON, занимает не более 1% общего времени, основная же доля затрачиваемого времени приходится на работу 1С со своей базой данных.

Напрашивающийся вывод – можно существенно ускорить загрузку информации, если сократить время записи в базу данных 1С. Управлять этим процессом в контексте 1С мы не можем, но «тонкая настройка» записи информации непосредственно в базу данных может дать определенные преимущества.

Технические ограничения такого решения:

* так как формат файловой базы данных 1С не документирован, база данных 1С должна быть SQL-типа (Microsoft SQL, Oracle, IBM DB2 или PostgreSQL);
* для исключения нарушений логической целостности информации на время загрузки базу данных целесообразно переводить в режим монопольного использования.

Проблемы при реализации прямой загрузки в базу данных:

1. искусственность именования объектов базы данных 1С (например, \_Reference19 вместо Продукция или \_Fld20RRef вместо КодКомпании);
2. (по информации из интернет) одна и та же конфигурация 1С при развертывании на разных серверах может генерировать разные имена для одних и тех же логических объектов базы данных;
3. в таблицах базы данных наряду с базовой информацией могут записываться служебные данные, например, итоговые остатки для каждого товара и склада на каждый месяц в Регистрах накопления.

Первые две проблемы могут быть решены вызовом в 1С специальной функции ПолучитьСтруктуруХраненияБазыДанных, которая предоставляет информацию о физической структуре базы данных в проекции на объекты 1С. Более того, существует ряд готовых обработок для 1С, позволяющих в наглядной форме получить структуру базы данных, например, <http://assorti-it.ru/public/568938/> .

Вторая проблема может быть решена либо проведением ряда экспериментов по добавлению данных в рамках 1С с одновременным мониторингом изменений в базе данных, либо запуском режимов пересчета в 1С после загрузки данных.

#### Пример кода

C#

var products = (new ИсходныеДанные\_Генератор()).ПолучитьПродукцию().ToArray();

var companies = products.Select(p => p.Компания).Distinct();

using (SqlConnection con = new SqlConnection(@"Server=.\sqlexpress;Database=Test1S;Trusted\_Connection=True;"))

{

con.Open();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.CommandText =

@"INSERT INTO \_Reference7 (\_IDRRef,\_Marked,\_PredefinedID,\_Code,\_Description)

VALUES (@\_IDRRef,@\_Marked,@\_PredefinedID,@\_Code,@\_Description)";

cmd.Parameters.Add("@\_IDRRef", SqlDbType.Binary, 16);

cmd.Parameters.AddWithValue("@\_Marked", new byte[1]);

cmd.Parameters.AddWithValue("@\_PredefinedID", new byte[16]);

cmd.Parameters.Add("@\_Code", SqlDbType.NVarChar, 9);

cmd.Parameters.Add("@\_Description", SqlDbType.NVarChar, 25);

foreach (var c in companies)

{

cmd.Parameters[0].Value = c.\_IDRRef;

cmd.Parameters[3].Value = c.Код.ToString("D9");

cmd.Parameters[4].Value = c.Наименование;

cmd.ExecuteNonQuery();

}

cmd.CommandText =

@"INSERT INTO \_Reference19 (\_IDRRef,\_Marked,\_PredefinedID,\_Code,\_Description,\_Fld20RRef)

VALUES (@\_IDRRef,@\_Marked,@\_PredefinedID,@\_Code,@\_Description,@\_Fld20RRef)";

cmd.Parameters.Add("@\_Fld20RRef", SqlDbType.Binary, 16);

foreach (var p in products)

{

cmd.Parameters[0].Value = p.\_IDRRef;

cmd.Parameters[3].Value = p.Код.ToString("D9");

cmd.Parameters[4].Value = p.Наименование;

cmd.Parameters[5].Value = p.Компания.\_IDRRef;

cmd.ExecuteNonQuery();

}

}

##### Особенности кода:

1. Для заполнения используется база данных Microsoft SQL Express, код для работы с другими базами данных будет аналогичным.

##### Оценка быстродействия

Неожиданный результат – время загрузки в среднем в 1,5 раза больше, чем при загрузке средствами 1С.

Вывод: разработчики 1С хорошо оптимизируют работу с базой данных.

### Оптимизация для Microsoft SQL Server

В Microsoft SQL Server, начиная с версии 2008, предусмотрен специальный оператор для импорта файлов данных BULK INSERT [[[13]](#endnote-9)]. ADO.Net позволяет вообще не формировать файлы загрузки, а передавать массовые данные непосредственно на сервер – для этого используется класс SqlBulkCopy [[[14]](#endnote-10)]. Пример его использования:

using (SqlConnection con =

new SqlConnection(@"Server=.\sqlexpress;Database=Test1S;Trusted\_Connection=True;"))

{

con.Open();

SqlBulkCopy sqlBulkCopy = new SqlBulkCopy(con, SqlBulkCopyOptions.TableLock, null);

sqlBulkCopy.DestinationTableName = "\_Reference7";

sqlBulkCopy.WriteToServer(данные.Компании);

sqlBulkCopy.DestinationTableName = "\_Reference19";

sqlBulkCopy.WriteToServer(данные.Продукция);

}

Здесь данные – сформированный в памяти набор исходных данных (технически – типизированный DataSet, структура которого эквивалента структуре таблиц-получателей в Microsoft SQL Server).

Использование указанной техники даже без оптимизации настроек массового копирования и без использования возможностей многопоточной обработки данных позволило сократить время загрузки на два порядка – до 0,6 сек.

Примечание: формирование исходных данных полностью в памяти не является единственно возможной техникой использования SqlBulkCopy. Источник данных может быть любой, будь то XML-файл, CSV-файл или другая СУБД, например, Oracle или MySQL. Достаточно получить из источника данные в виде объекта DataTable или реализовать интерфейс IDataReader поверх методов доступа к данным.

### Оптимизация для Oracle

В Oracle, начиная с версии 11g Release 2, также реализована возможность массовой загрузки данных [[[15]](#endnote-11)]. Соответствующая версия Oracle Data Provider for .NET содержит класс OracleBulkCopy [[[16]](#endnote-12)], функционально эквивалентный описанному выше SqlBulkCopy. Можно ожидать, что использование этого класса даст соизмеримый рост скорости загрузки (к сожалению, из-за недоступности Oracle экспериментальная проверка не проводилась).

# Выводы

Сравнительные результаты рассмотренных выше способов загрузки внешних данных в 1С сведены в таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Способ загрузки | Время, сек | Сложность кода 1С | Сложность кода C# | Рекомендации |
| [Импорт текстовых файлов – чтение всего файла в память](#_Пример_кода_1_1) | 57 | 3 | - | Для малых объемов данных |
| [Импорт текстовых файлов – построчное чтение файла](#_Пример_кода_2_1) | 58 | 3 | - | Для больших объемов данных |
| [Импорт XML-файлов – чтение всего файла в память, прямой доступ к тэгам и атрибутам](#_Пример_кода_1) | 59 | 2 | - | Для малых объемов данных |
| [Импорт XML-файлов – последовательное чтение файла, прямой доступ к тэгам и атрибутам](#_Пример_кода_2) | 59 | 3 | - | Для больших объемов данных |
| [Импорт XML-файлов – чтение всего файла в память, десериализация 1С-объектов с помощью XDTO](#_Пример_кода_3) | 79 | 2 | - | Для малых объемов данных |
| [Импорт JSON-файлов](#_Импорт_JSON-файлов) | 63 | 2 | - |  |
| [Использование встроенного механизма доступа к внешним ODBC-данным](#_Использование_встроенного_механизма_1) | 89 | 2 | - |  |
| [Вызов пользовательского COM-объекта](#_Вызов_пользовательского_COM-объекта) | 61 | 1 | 3 |  |
| [Загрузка из Microsoft SQL Express через ADO](#_Вызов_COM-объекта_ADO) | 63 | 3 | - |  |
| [Вызов 1С в качестве COM-сервера](#_Вызов_1С_как) | ∞ | - | 4 |  |
| [Прямая загрузка в базу данных 1С – Microsoft SQL Express без оптимизации](#_Общее_решение) | 88 | - | 2 |  |
| [Прямая загрузка в базу данных 1С – Microsoft SQL Express с оптимизацией (SqlBulkCopy)](#_Оптимизация_для_Microsoft) | 0,6 | - | 1 |  |

Примечание: Для тестирования использовался искусственно генерируемый набор данных: 50 компаний по 100 видов продукции в каждой – итого = 50 + 5000 = 5050 записей.

На основании представленных в таблице данных можно сделать вывод о предпочтительности использования для загрузки исходных данных последнего способа – прямой загрузки в базу данных 1С с оптимизацией через BulkCopy.

Если прямую загрузку в базу данных использовать нельзя, наиболее простым и быстрым способом будет передача информации через текстовые файлы.

В случае, если потребуется более «тонкое» управление процессом синхронизации, например, на основе событийной модели, удачным решением может быть использование пользовательского COM-объекта.

# Ссылки

1. Обеспечить совпадение физической структуры промежуточной базы данных с базой 1С - отдельная техническая проблема в силу закрытости форматов 1С. [↑](#footnote-ref-1)
2. Средства интеграции 1С – <http://v8.1c.ru/overview/Term_000000581.htm> [↑](#endnote-ref-1)
3. Прямой доступ к базе данных официально 1С не поддерживается и запрещен лицензионным соглашением 1С: «…не совершать и не допускать совершения третьими лицами следующих действий … доступ к информационной базе ПРОГРАММНОГО ПРОДУКТА и построение систем на основе ПРОГРАММНОГО ПРОДУКТА с помощью средств и технологических решений, не предусмотренных в сопроводительной документации». [↑](#footnote-ref-2)
4. Реализовано в версии 1С 8.3.6.1977 и выше [↑](#footnote-ref-3)
5. Универсальный механизм обмена данными – <http://v8.1c.ru/overview/Term_000000314.htm> [↑](#endnote-ref-2)
6. Реализовано в версии 1С 8.2.14.533 и выше [↑](#footnote-ref-4)
7. Внешние источники данных 1С – <http://howknow1c.ru/programmirovanie-1c/vneshnie-istochniki-dannyh-1s.html> [↑](#endnote-ref-3)
8. Технология создания внешних компонент 1С – <http://its.1c.ru/db/metod8dev#content:3221:hdoc:_input:native%20api> [↑](#endnote-ref-4)
9. Exposing an Enumerator from Managed Code to COM – <https://limbioliong.wordpress.com/2011/10/28/exposing-an-enumerator-from-managed-code-to-com/> [↑](#endnote-ref-5)
10. Документация ADO: MSDN. ADO API Reference – <https://docs.microsoft.com/en-us/sql/ado/reference/ado-api/ado-api-reference> [↑](#endnote-ref-6)
11. Синтаксис и формат строк соединения – <https://www.connectionstrings.com> [↑](#endnote-ref-7)
12. Режим MARS – <https://msdn.microsoft.com/ru-ru/library/cfa084cz(v=vs.110).aspx> [↑](#endnote-ref-8)
13. BULK INSERT (Transact-SQL) – <https://msdn.microsoft.com/ru-ru/library/ms188365.aspx> [↑](#endnote-ref-9)
14. Класс SqlBulkCopy – <https://msdn.microsoft.com/ru-ru/library/system.data.sqlclient.sqlbulkcopy(v=vs.110).aspx> [↑](#endnote-ref-10)
15. Bulk Processing with BULK COLLECT and FORALL – <http://www.oracle.com/technetwork/issue-archive/2012/12-sep/o52plsql-1709862.html> [↑](#endnote-ref-11)
16. OracleBulkCopy Class – <https://docs.oracle.com/cd/E17666_01/doc/win.112/e17357/OracleBulkCopyClass.htm> [↑](#endnote-ref-12)