# Курсовой проект “Олимпийские игры”

Пояснительная записка курсового проекта содержит 27 страницы, 21 рисунков, 3 источника литературы, 4 приложения.

БИБЛИОТЕКА, БАЗА ДАННЫХ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, SQL, T-SQL, MICROSOFT SQL SERVER.

Основными целями курсового проекта являются: проектирование базы данных «Олимпийские игры».

Пояснительная записка состоит из введения, пяти разделов, заключения.

Во введении представлена общая информация, дающая представление о предстоящей работе, определены цели.

В первом разделе рассматривается постановка задач.

Во втором разделе рассматривается разработка модели базы данных.

В третьем разделе описывается создание процедур для экспорта и импорта.

В четвертом разделе представляется реализация технологии.

В пятом разделе представлено руководство пользователя.

В заключении представлены итоги курсового проектирования и задачи, которые были решены в ходе проектирования и разработки базы данных.

В приложении вставлены листинги когда программы и процедур.

Содержание

[Курсовой проект “Олимпийские игры” 1](#_Toc501021828)

[Введение 3](#_Toc501021829)

[1. Постановка задачи 4](#_Toc501021830)

[1.1 Обзор и анализ прототипов 4](#_Toc501021831)

[1.2 Актуальность решаемой задачи 5](#_Toc501021832)

[2. Разработка модели базы данных 6](#_Toc501021833)

[3. Описание процедур импорта и экспорта 10](#_Toc501021834)

[3.1. Процедура импорта 10](#_Toc501021835)

[3.2. Процедура экспорта 10](#_Toc501021836)

[4. Технология SSIS (SQL Server Intergration Services) 11](#_Toc501021837)

[5. Руководство пользователя 14](#_Toc501021838)

[Заключение 16](#_Toc501021839)

[Список использованных источников 17](#_Toc501021840)

[ПРИЛОЖЕНИЕ А 18](#_Toc501021841)

[ПРИЛОЖЕНИЕ Б 20](#_Toc501021842)

[ПРИЛОЖЕНИЕ В 23](#_Toc501021843)

[ПРИЛОЖЕНИЕ Г. 26](#_Toc501021844)

# Введение

Современное общество невозможно представить без компьютера. Компьютеры настолько широко и глубоко внедрились в нашу жизнь, что очень трудно назвать какую-либо сферу деятельности человека, где бы они ни использовались. В связи с этим серьезные требования предъявляются как к аппаратной части современных компьютеров, так и к используемому программному обеспечению. В основном именно программное обеспечение, или, иными словами, программные продукты, обеспечивают возможность широкого использования компьютеров. Стоит лишь переустановить программное обеспечение компьютера или добавить какой-либо новый программный продукт, и можно будет решать на этом компьютере совершенно новые задачи.

Следовательно, используемые программные продукты должны соответствовать определенным критериям, обеспечивающим надежность работы компьютера и удобство работы пользователя.

В соответствии с заданием курсового проектирования следует создать базу данных (БД) «Олимпийские игры» с использование программного проекта Microsoft SQL Server.

Сегодня трудно себе представить сколько-нибудь значимую информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных. Концепции и технологии БД складывались постепенно и всегда были тесно связаны с развитием систем автоматизированной обработки информации.

Проектирование базы данных «Олимпийские игры» ведется для просмотра и учета различной информации.

# Постановка задачи

# Обзор и анализ прототипов

На сегодняшний день существует несколько программных продуктов для автоматизации работы с олимпийскими играми.

Проанализировав некоторые из сайтов с результатами олимпийских игр, в качестве прототипа был выбран сайт http://olympteka.ru. В отличие от других сайтов он показался мне наиболее лаконичным и содержательным, к тому же располагающим к быстрой выборке.

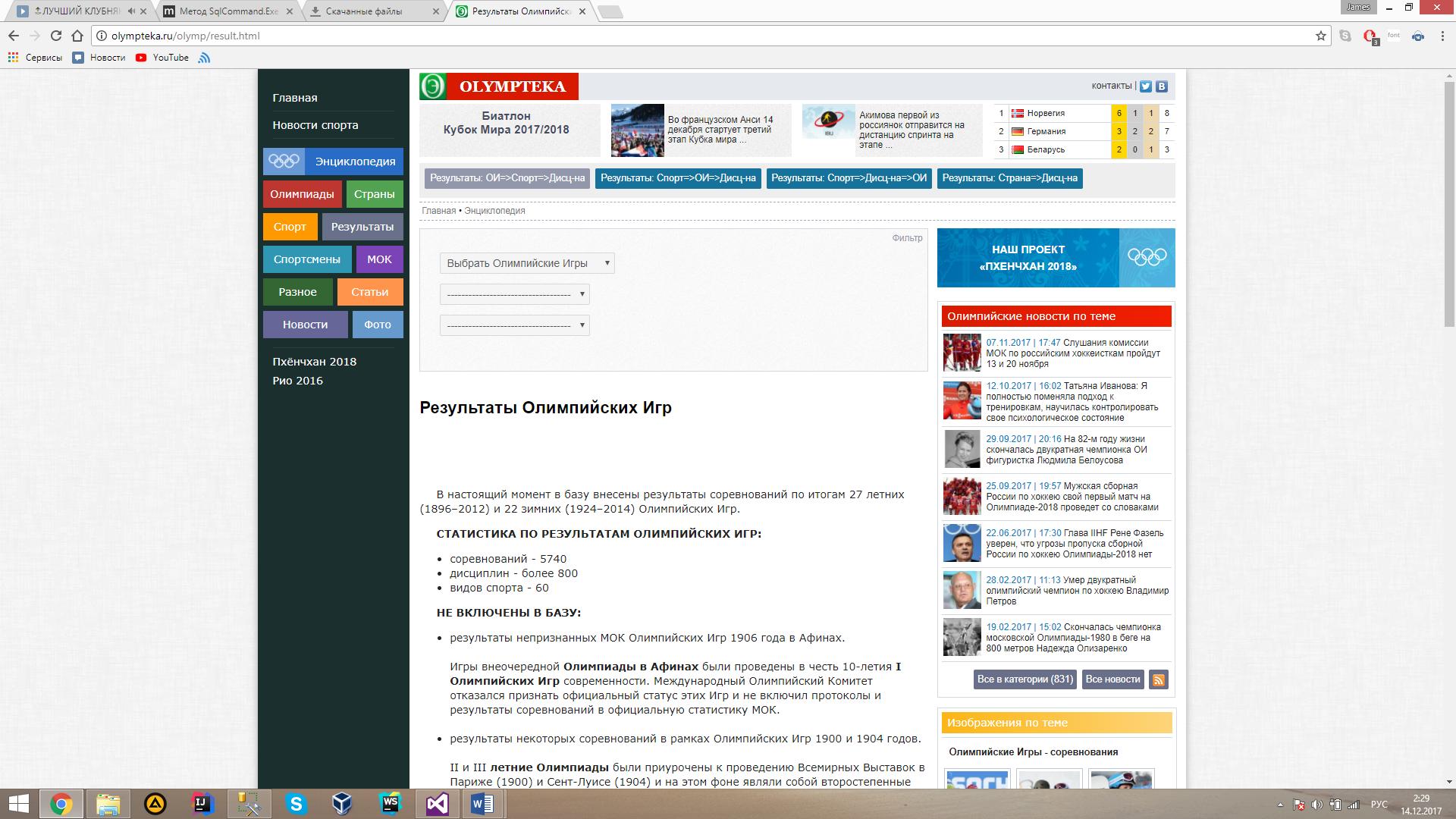


Рисунок 1 − Интерфейс начальной страницы сайта

Принимая во внимание представленный на рисунке 1 прототип, а также учитывая то, что был дополнительно произведен анализ ряда программ для подбора туристических туров, можно сказать о следующих сходствах программных средств:

* наличие бокового меню;
* поле поиска олимпийских игр по заданным критериям;
* подбор различной контекстной информации;
* другие возможности сайта.

Существует ряд аналогов, превосходящих мой программный продукт по объему функционала, однако приложение OlympicGames Database является десктопным и не требует доступа к Интернету для просмотра и администрирования различной информации.

# Актуальность решаемой задачи

Персональный компьютер и интернет, их доступность и надежность, способствуют проникновению во все сферы общества новых информационных технологий. Эти технологии обеспечивают рост производительности в сфере услуг, в частности и в администрировании баз данных в режиме офлайн.

Сейчас уже трудно представить эффективную работу учета олимпийских игр без собственного сайта (приложения). Для олимпийских игр – это быстрый выход на клиентов. Основным требованием к программному продукту олимпийские игры является быстрота и корректность данных. В приложении для олимпийских игр нужна в первую очередь информация об актуальных в данный момент играх. Причем поисковая система должна быть четко организована, иметь возможность как целевого, так и расширенного поиска. Сама процедура поиска должна быть оптимизирована по времени.

# Разработка модели базы данных

* 1. **Проектирование базы данных**

Для хранения и предоставления данных пользователю была разработана архитектура базы данных, которая содержит в себе десять таблиц. Диаграмма базы данных представлена на рисунке 2.

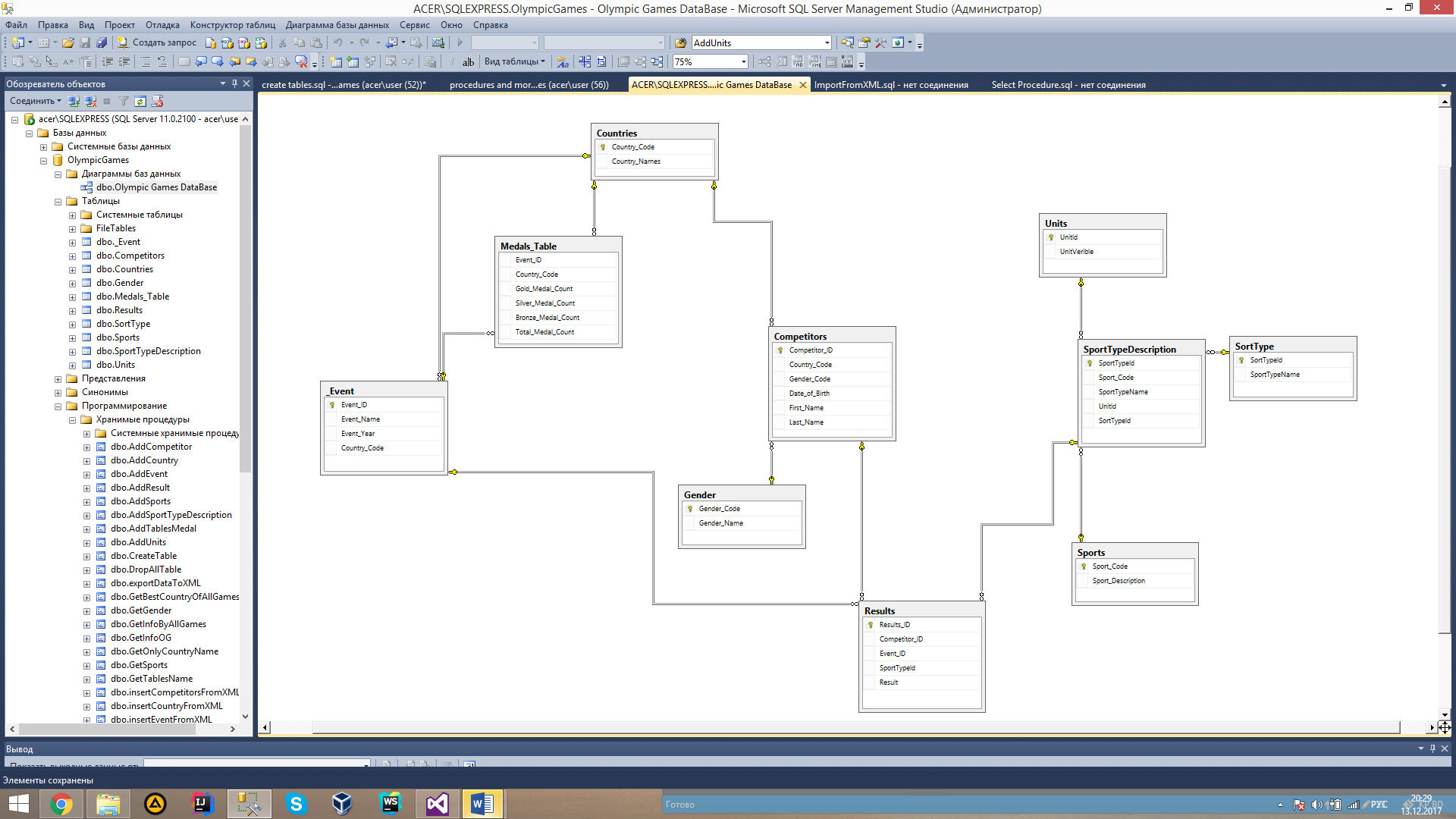


Рисунок 1. – Диаграмма базы данных

Таблица *Medals\_Table –* хранит данные медальном зачете по олимпийским играм.

Таблица *Results –* хранит результаты всех участников, по всем годам.

Таблица *Event –* хранит информацию названии и месте проведения олимпийских игр

Таблица *Countries –* содержит информацию о странах.

Таблица *Competitors­–* содержит информацию о участниках олимпийских игр.

Таблица *Gender –* содержит информацию о поле участника.

Таблица *Sports –* содержит информацию виде спорта.

Таблица *SportTypeDescription –* содержит информацию дисциплинах связанных с конкретным видом спорта.

Таблица *Units –* содержит информацию о еденицах измерения.

Таблица *SortType –* содержит информацию о типе сортировки.

* 1. **Резервное копирование и восстановление**

Резервное копирование базы данных будет осуществляться в папку Backup на диске D:\ каждый день начиная с первого декабря, в течении 30 дней и имя файла будет состоять из даты создания бэкапа.

Листинг исходного кода резервного копирования:

DECLARE @pathName NVARCHAR(512)

SET @pathName = 'D:\Backup\db\_backup\_' + Convert(varchar(8), GETDATE(), 112) + '.bak'

BACKUP DATABASE OlympicGames TO DISK = @pathName WITH NOFORMAT, NOINIT, NAME = N'db\_backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

Далее создаем файл расширения .bat, который и будет запускать скрипт резервного копирования.

Листинг исходного батника:

sqlcmd -S acer\SQLEXPRESS -E -i backup.sql

Создание задачи будем осуществлять с помощью планировщика заданий (запуск команды taskschd.msc).

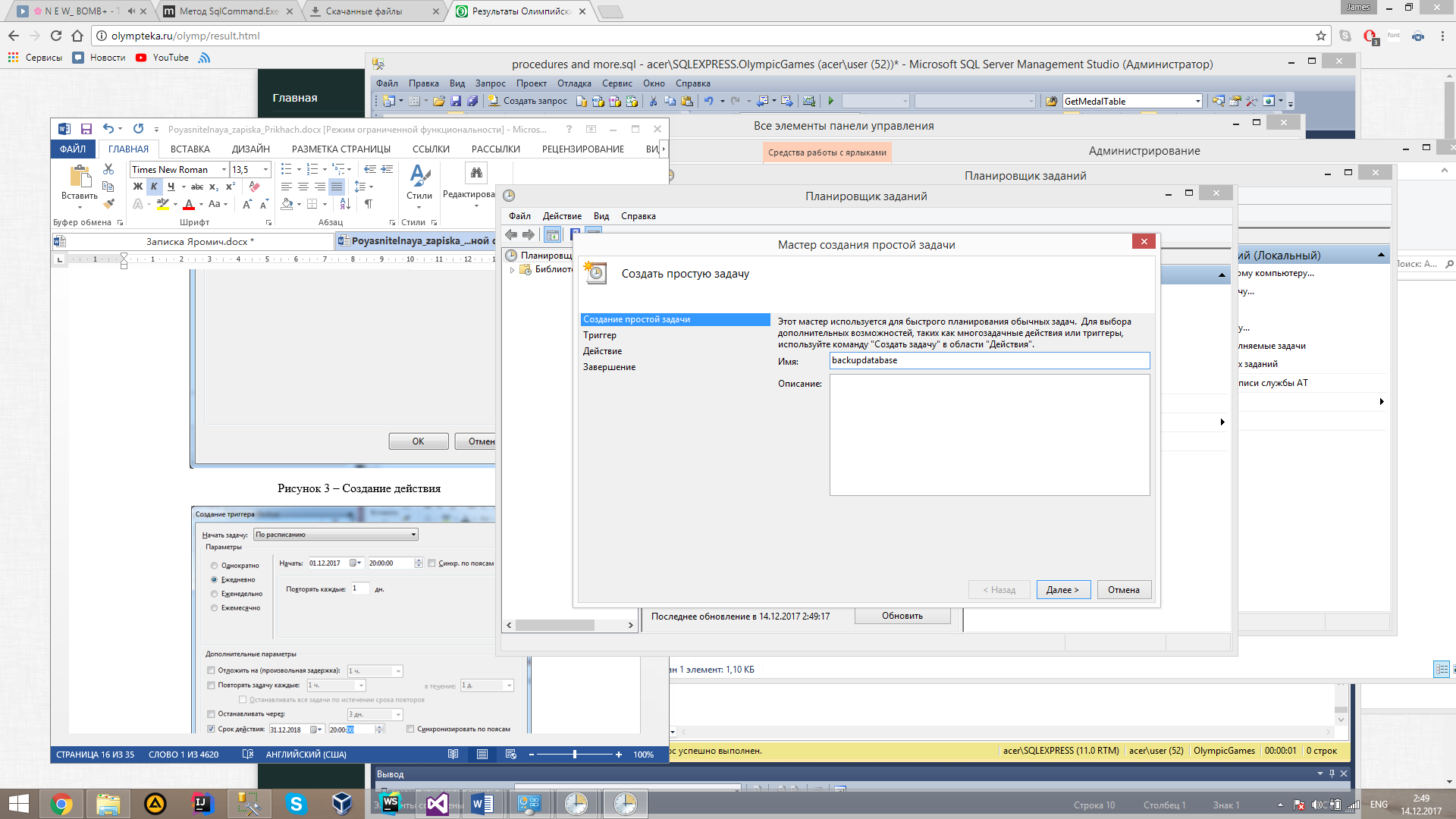


Рисунок 3 – Создание действия

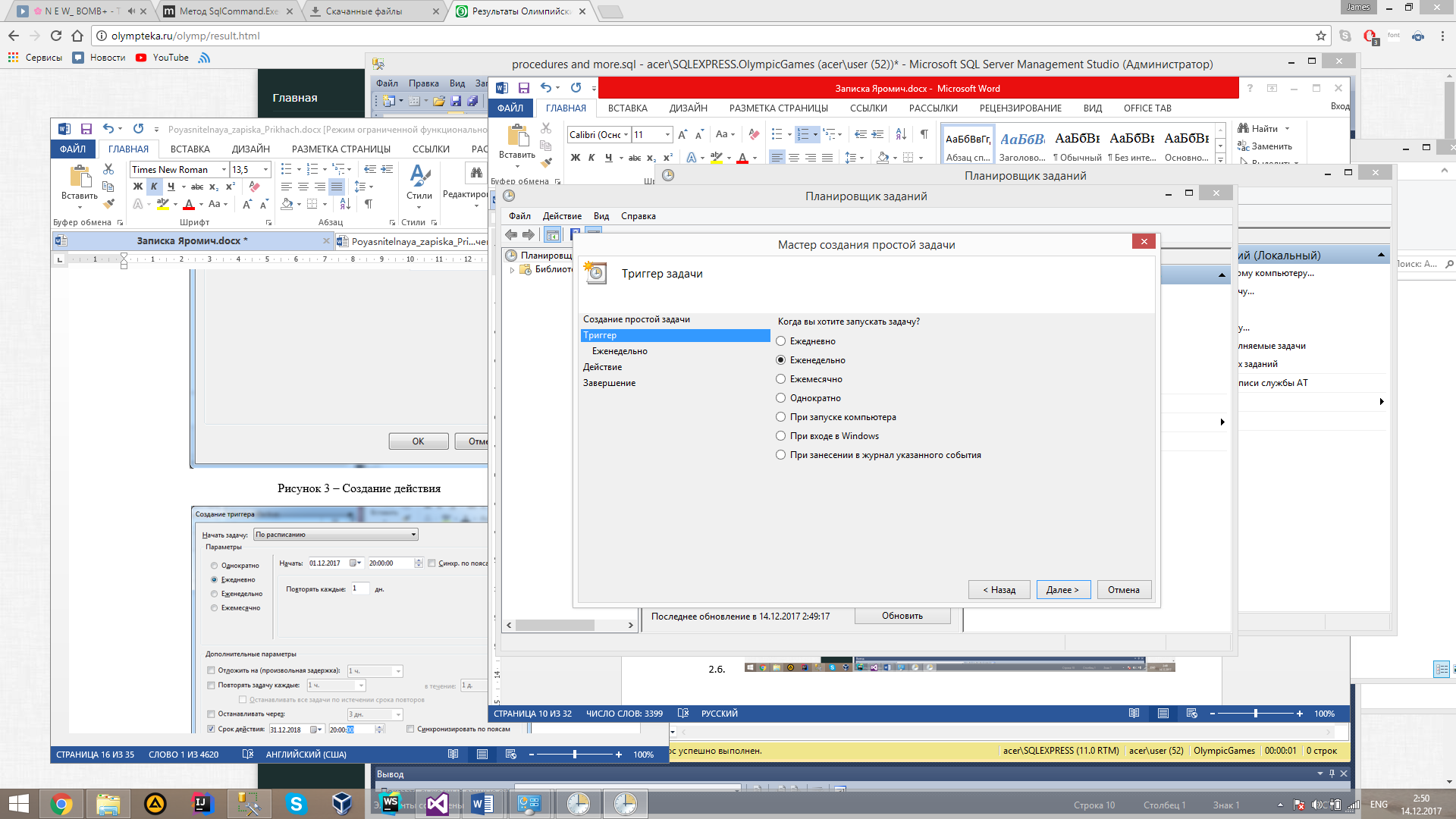


Рисунок 4 – Создание расписания

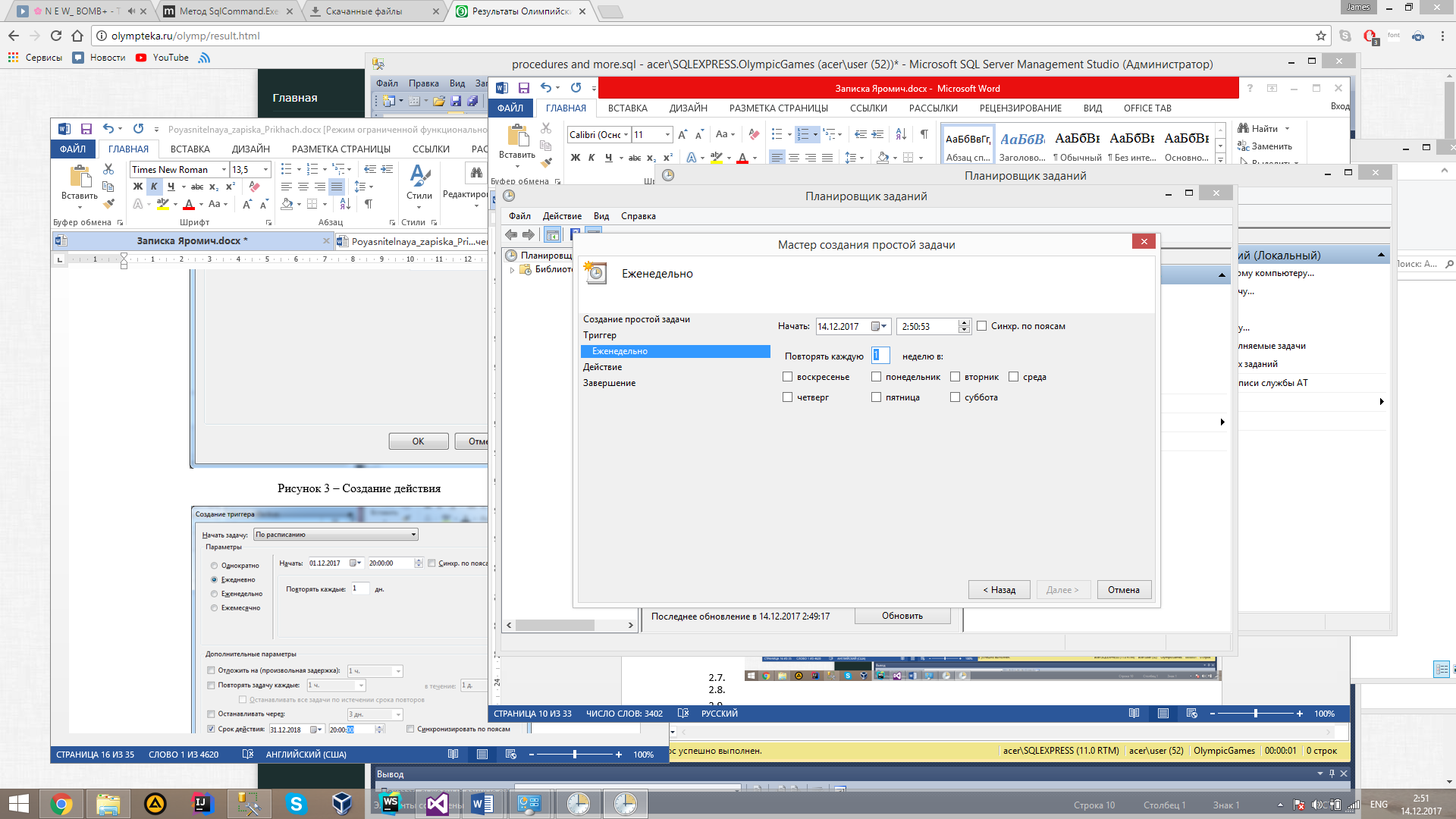


Рисунок 5 – Расписание(продолж.)

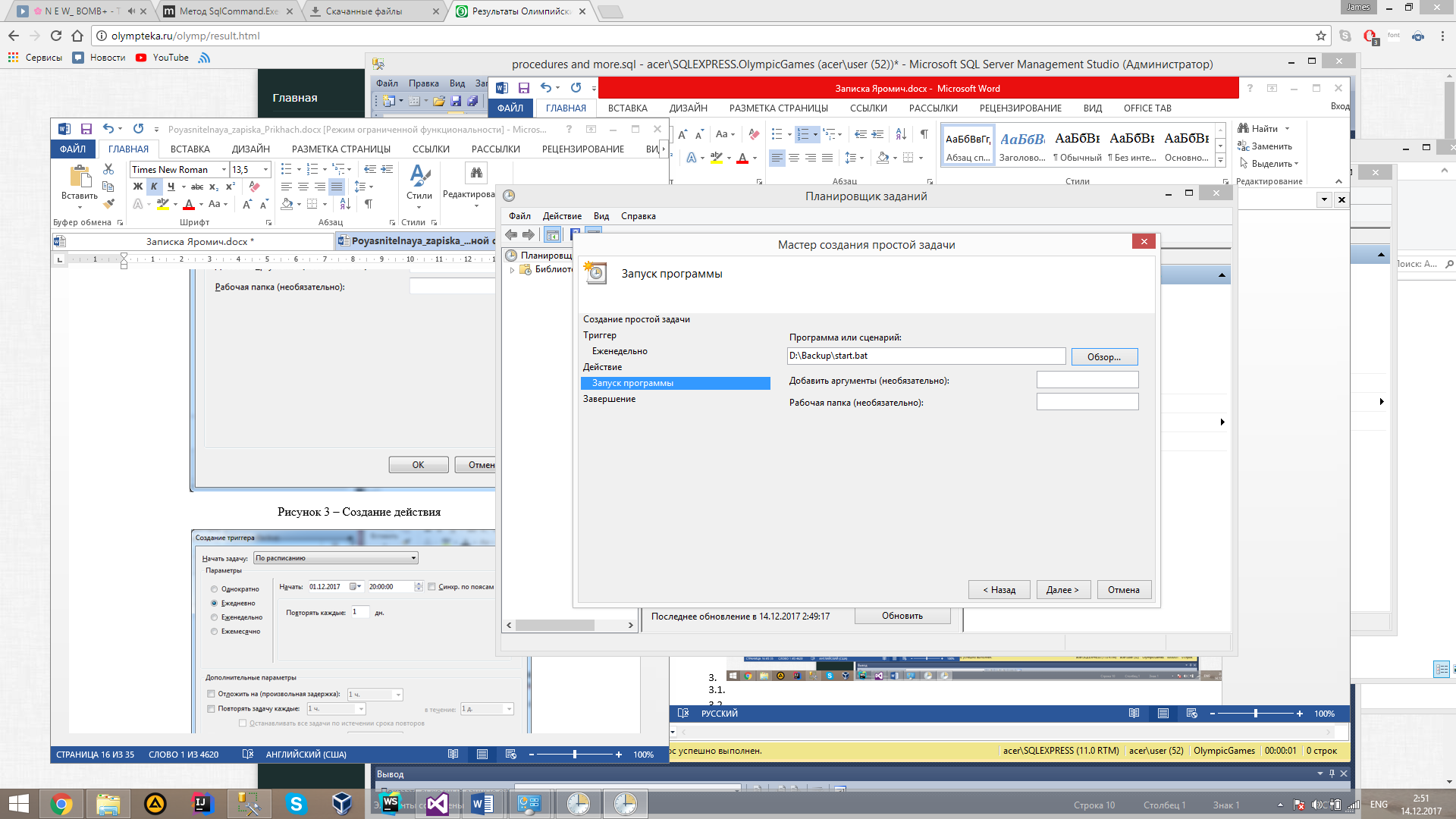


Рисунок 6 – Запуск программы.

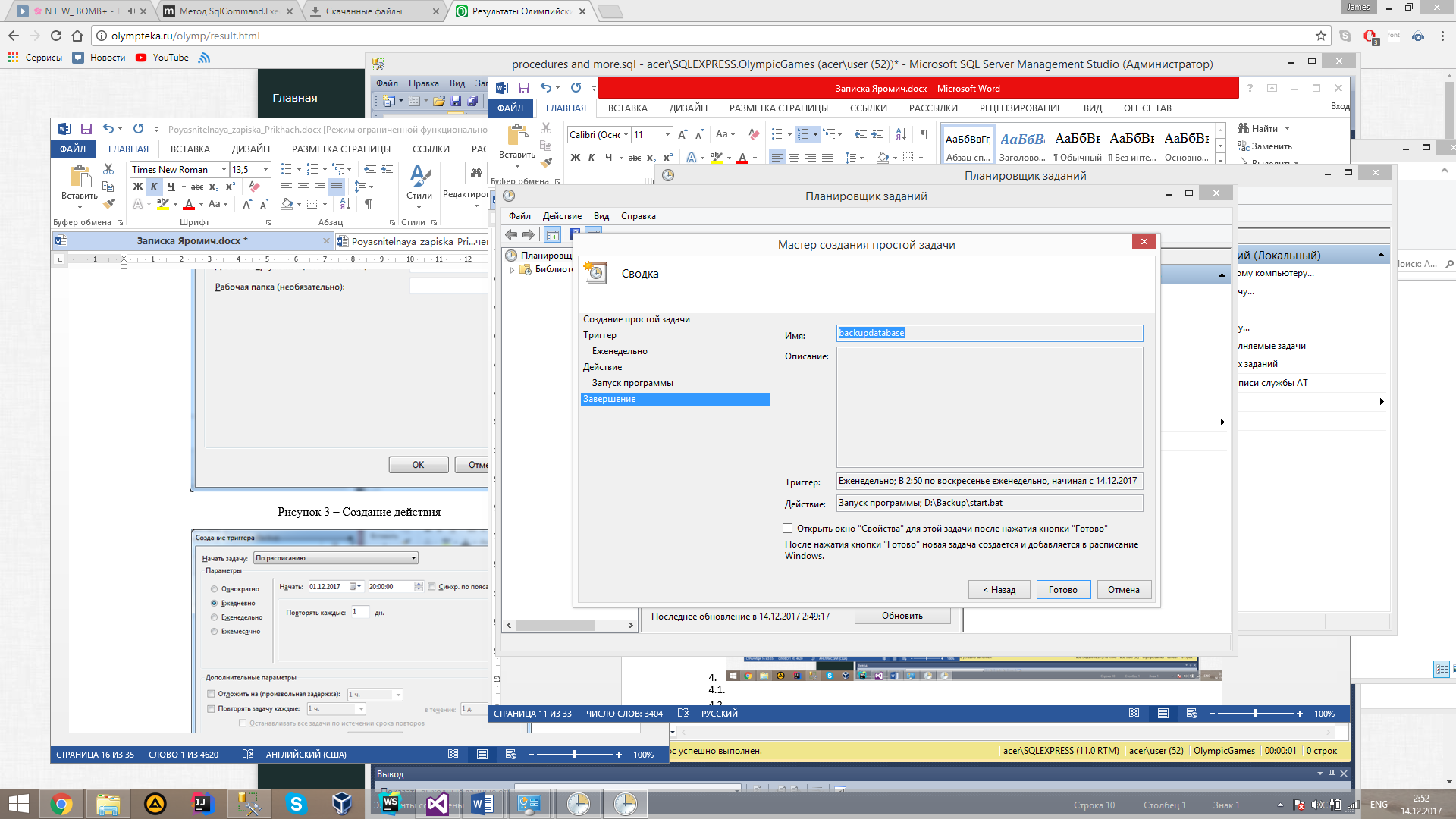


Рисунок 7 – Завершение

Восстановление базы данных можно осуществить с помощью выполнения задачи восстановления.

# Описание процедур импорта и экспорта

Для экспорта и импорта использовалось API разработанной Microsoft – OLE DB. OLE DB ( Object Linking and Embedding, Database) — набор интерфейсов, основанных на COM, которые позволяют приложениям обращаться к данным, хранимым в разных источниках информации или хранилищах данных с помощью унифицированного доступа. Импорт и экспорт производится в формат .xls и .xml согласно заданию курсового проекта.

## 3.1. Процедура импорта

Часть скрипта создания процедуры импорта представлен на рисунке 3.1.

declare @sql nvarchar(300)=

'SELECT

CAST(REPLACE(CAST(x AS VARCHAR(MAX)), ''encoding="utf-16"'', ''encoding="utf-8"'') AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';

Рисунок 3.1 – Процедура импорта

Полный листинг процедуры импорта находится в Приложении B. Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся

## 3.2. Процедура экспорта

Скрипт создания процедуры экспорта представлен на рисунке 3.2.

declare @sql nvarchar(500)=

'bcp "SELECT \* '+ 'FROM '+ @nametable +

' FOR XML PATH(''Order''), ROOT(''Root'')" queryout "'+@path+'" -S acer\SQLEXPRESS -d OlympicGames -w -T ';

EXEC xp\_cmdshell @sql;

Рисунок 3.2 – Процедура экспорта

Для экспорта используется .xml – файл, откуда экспортируются данные.

# Технология SSIS (SQL Server Intergration Services)

SSIS – это инструмент, который позволяет в удобном виде реализовать интеграцию, т.е. реализовать процесс переноса данных из одного источника в другой. Этот процесс иногда называют ETL (от англ. Extract, Transform, Load – дословно «извлечение, преобразование, загрузка»)

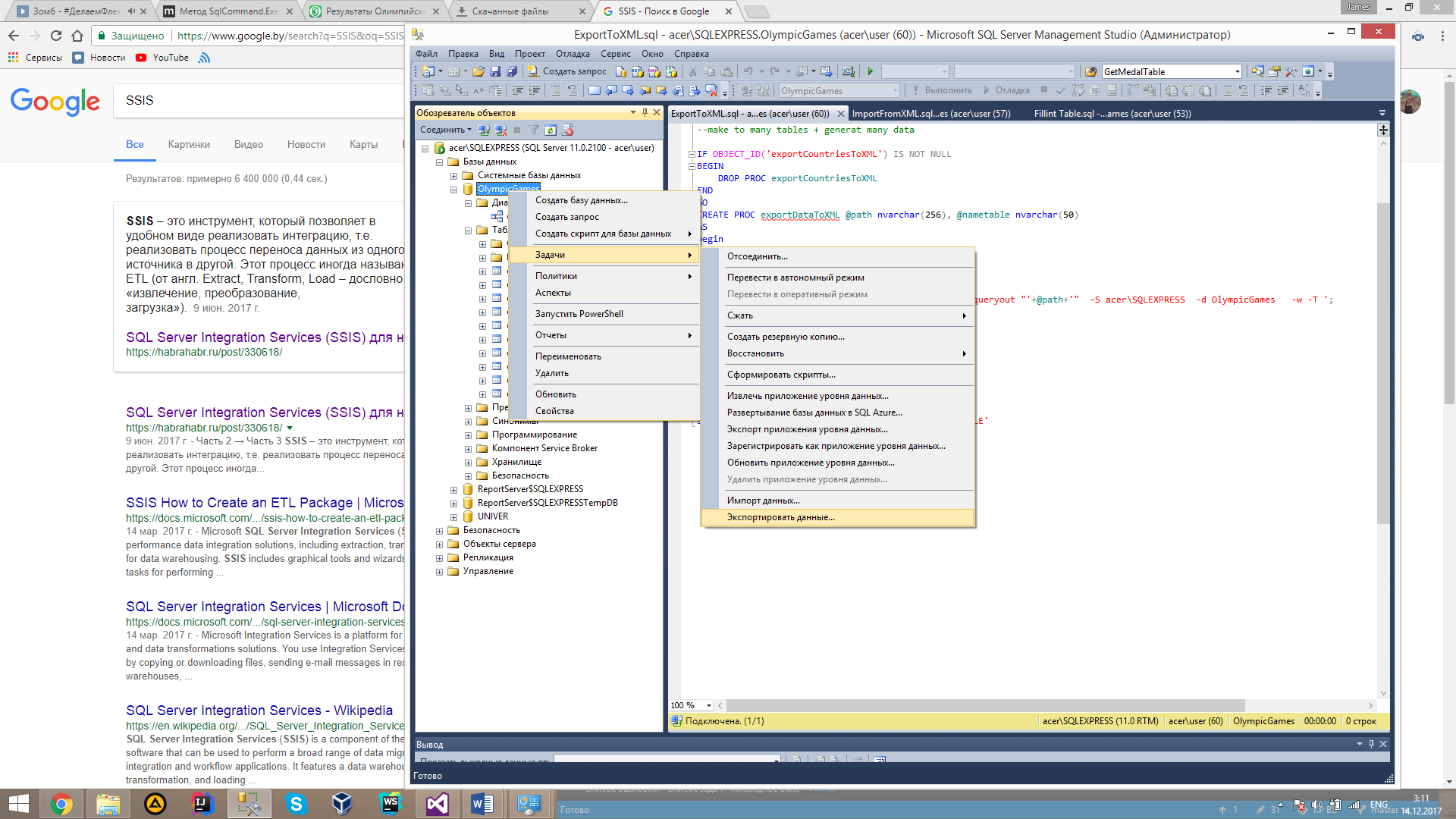


Рисунок 4.1. SSIS экспорт данных

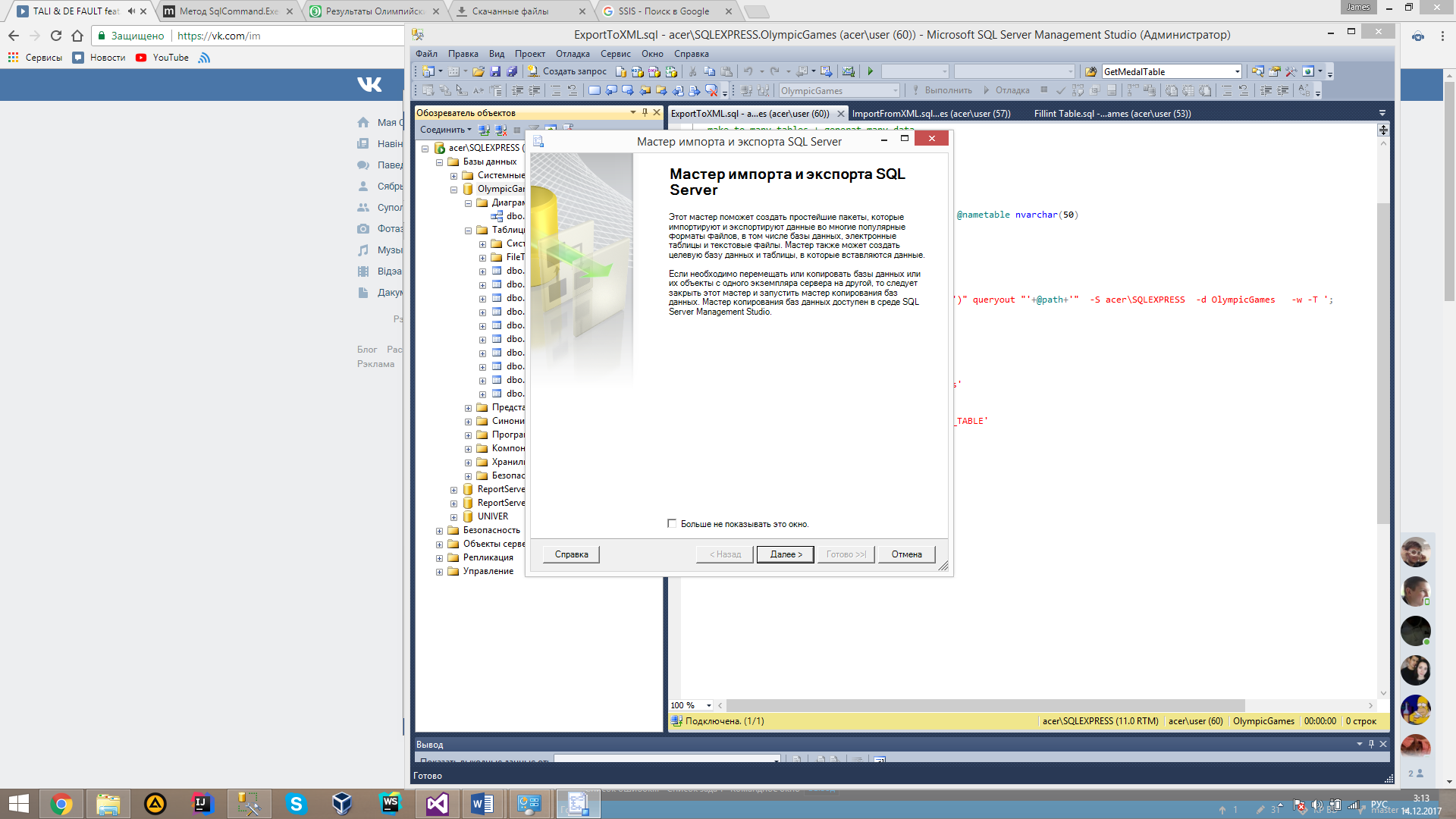


Рисунок 4.2 Мастер экспорта SQL Server

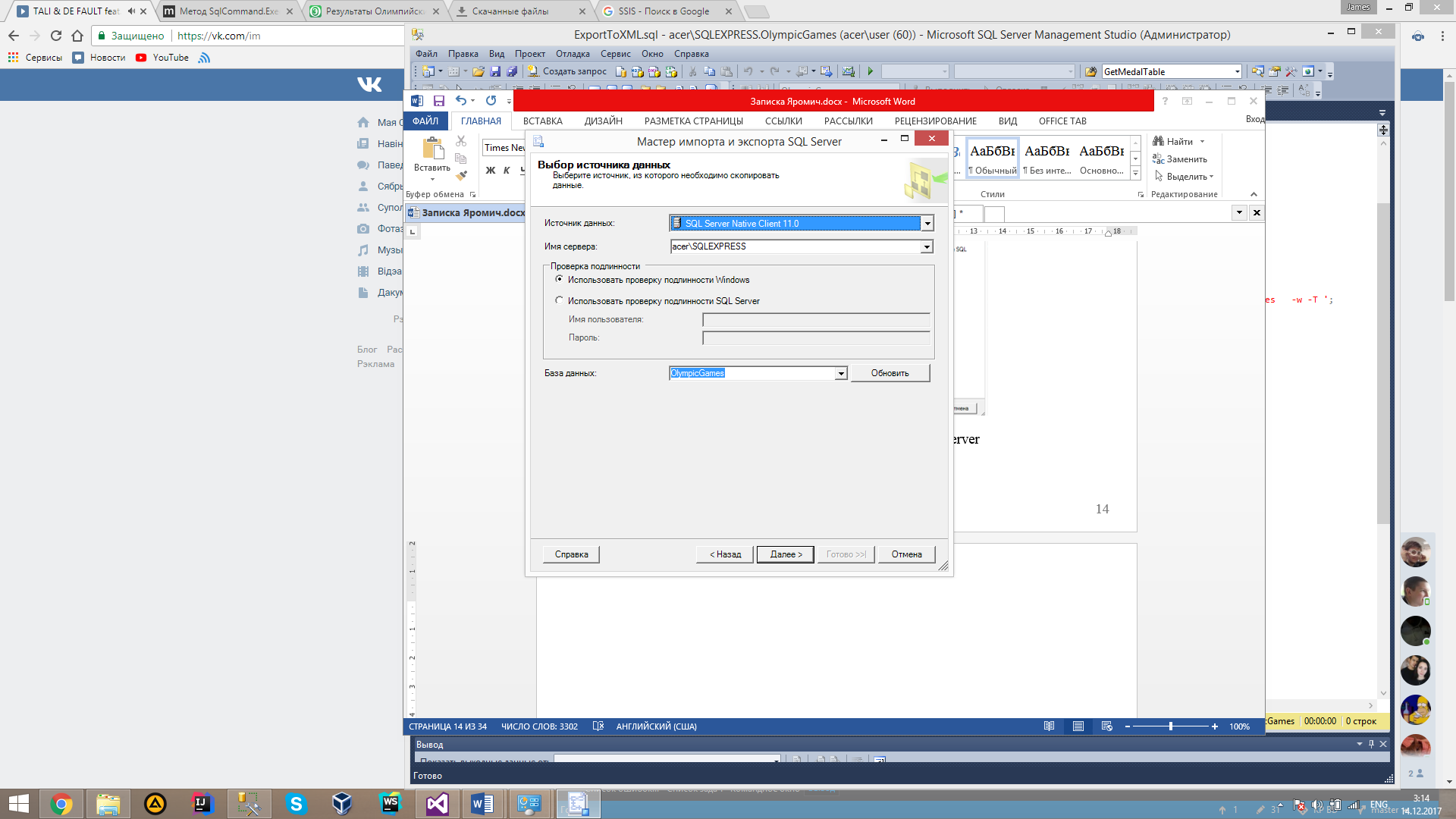


Рисунок 4.3 – Выбор источника данных

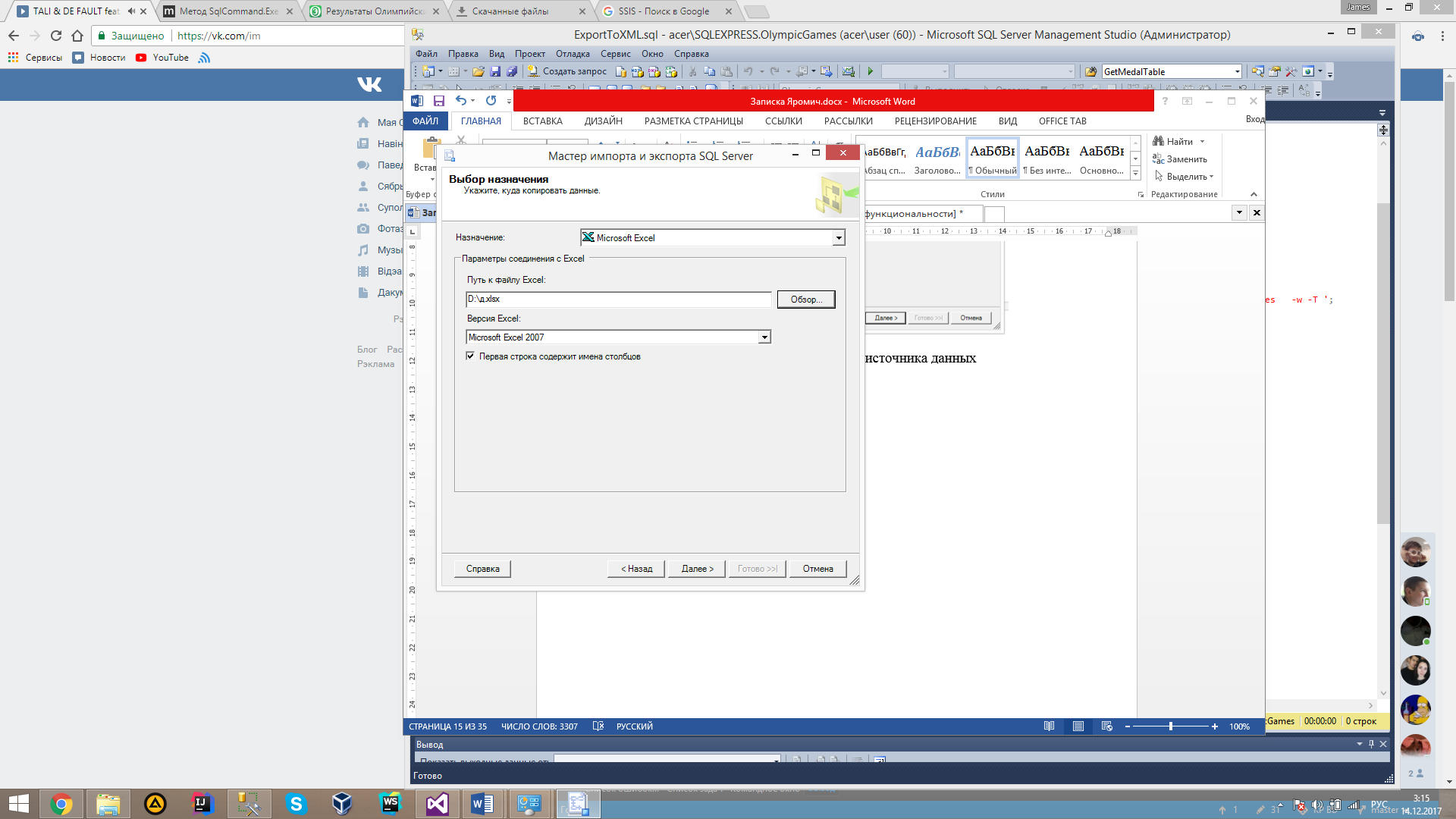


Рисунок 4.4 – Выбор источник экспорта

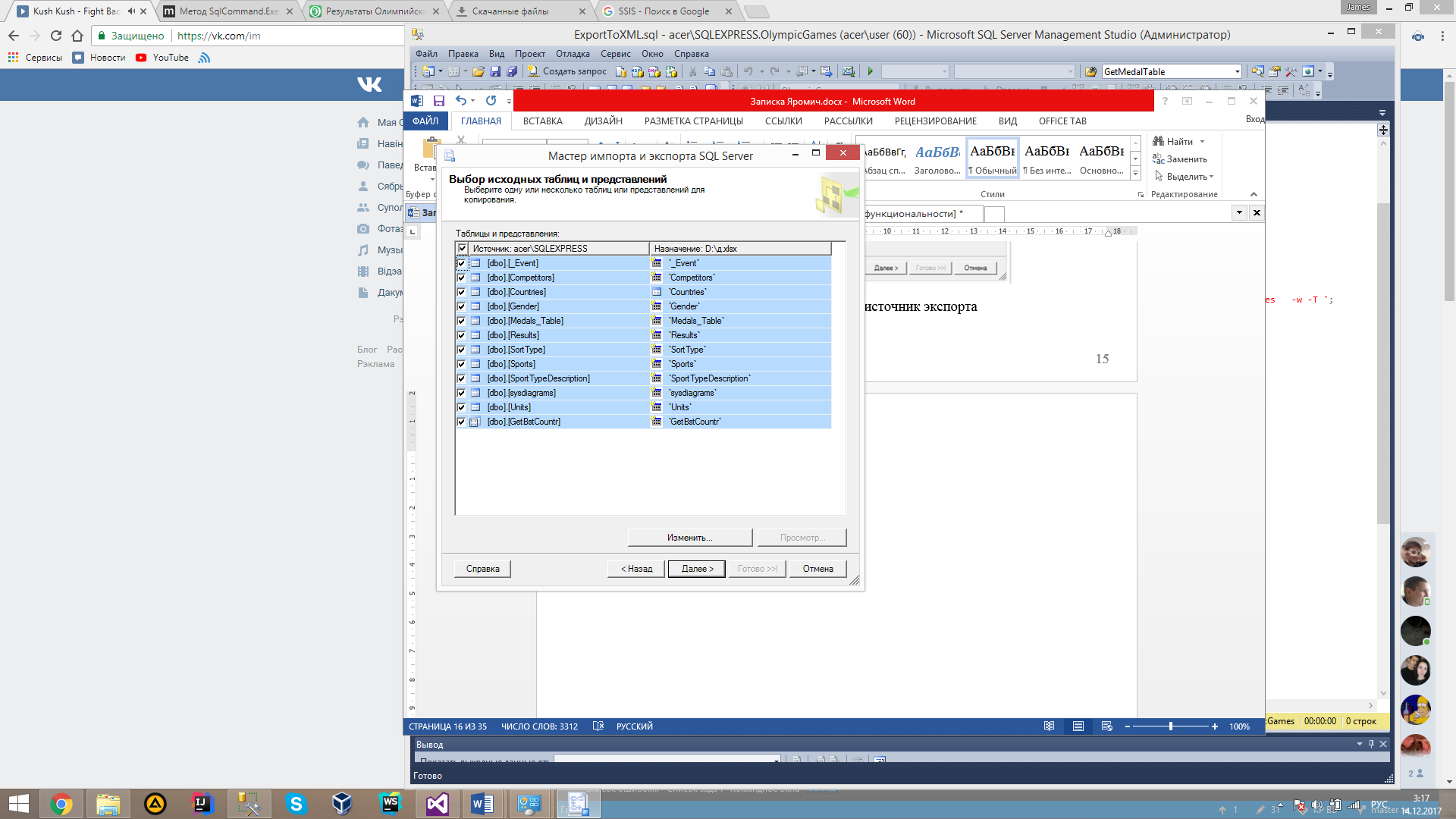


Рисунок 4.5 – Выбор информации для экспорта.

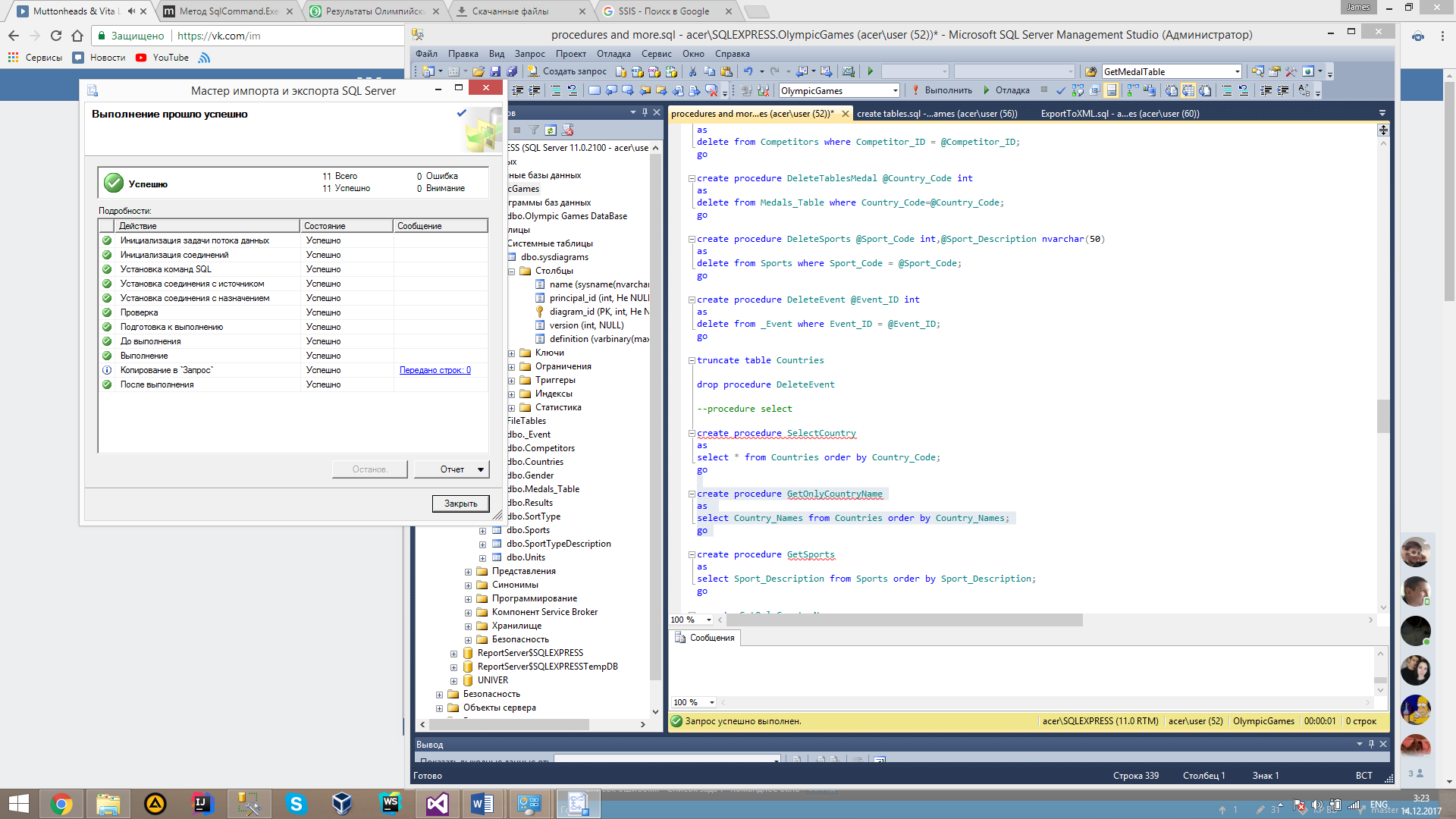


Рисунок 4.6. Результат обработки кастомного запроса

1. Описание процедур для работы с базой данных

Весь доступ к данным осуществляется

* 1. Процедура добавления страны в базу данных

create procedure AddCountry @Country\_Names nvarchar(50)

as begin

declare @Par nvarchar(50);

set @Par = (Select Country\_Names

from Countries

where Country\_Names = @Country\_Names);

if @Par is null

begin

insert into Countries(Country\_Names)

values (@Country\_Names);

end

end

go

Процедура принимает в качестве параметра название страны и добавляет ее в базу данных с предварительной проверкой.

* 1. Процедура добавления результата

create procedure AddResult @First\_Name nvarchar(50),

@Last\_Name nvarchar(50),

@Event\_Name nvarchar(50),

@SportTypeName nvarchar(50),

@Result nvarchar(50)

as begin

declare @CompId int;

set @CompId = ( select Competitor\_ID from Competitors where First\_Name = @First\_Name and Last\_Name = @Last\_Name );

declare @EvenId int;

set @EvenId = ( select Event\_ID from \_Event where Event\_Name = @Event\_Name);

declare @SportTypeId int;

set @SportTypeId = ( select SportTypeId from SportTypeDescription where SportTypeName = @SportTypeName);

insert into Results(Competitor\_ID,Event\_ID,SportTypeId,Result)

values (@CompId,@EvenId,@SportTypeId,@Result);

end

go

* 1. Процедура добавления участника.

create procedure AddCompetitor @Country\_Names nvarchar(50),

@Gender\_Name nvarchar(50),

@Date\_of\_Birth date,

@First\_Name nvarchar(50),

@Last\_Name nvarchar(50)

as begin

declare @Par nvarchar(50);

declare @SecondPar nvarchar(50);

declare @GendCode int;

declare @CountryCode int;

set @GendCode = ( Select Gender.Gender\_Code

from Gender

where Gender\_Name = @Gender\_Name

);

set @CountryCode = ( Select Country\_Code

from Countries

where Country\_Names = @Country\_Names

);

begin

insert into Competitors(Country\_Code,Gender\_Code,Date\_of\_Birth,First\_Name,Last\_Name)

values (@CountryCode,@GendCode,@Date\_of\_Birth,@First\_Name,@Last\_Name);

end

end

go

Листинги других процедур находятся в приложения Б.

Практически все процедуры добавления действуют по одному принципу.

Получаем значения для добавление в таблицу, если это нужно, проверяем есть ли в таблице данные и только тогда добавляем в базу данных.

# Руководство пользователя

Поскольку в базе данных нет разделения на пользователей, то данное руководство является общим.

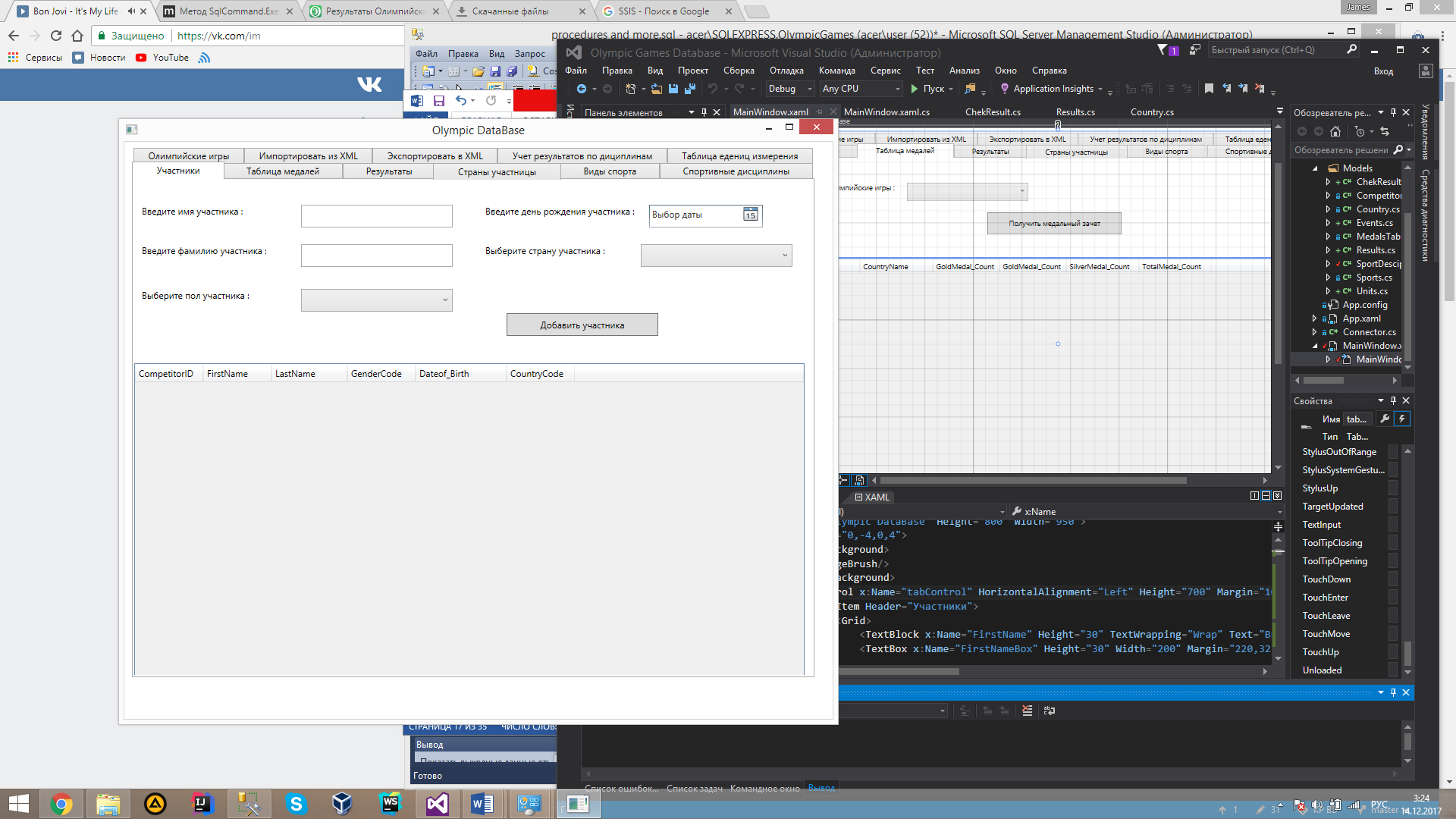


Рисунок 5.1.1 – Панель приложения

Пользователь программы может перейти по любой из вкладок и выбрать ему нужную информацию для изменения или просмотра.

Так же, доступна сортировка по критериям и просмотр данных.

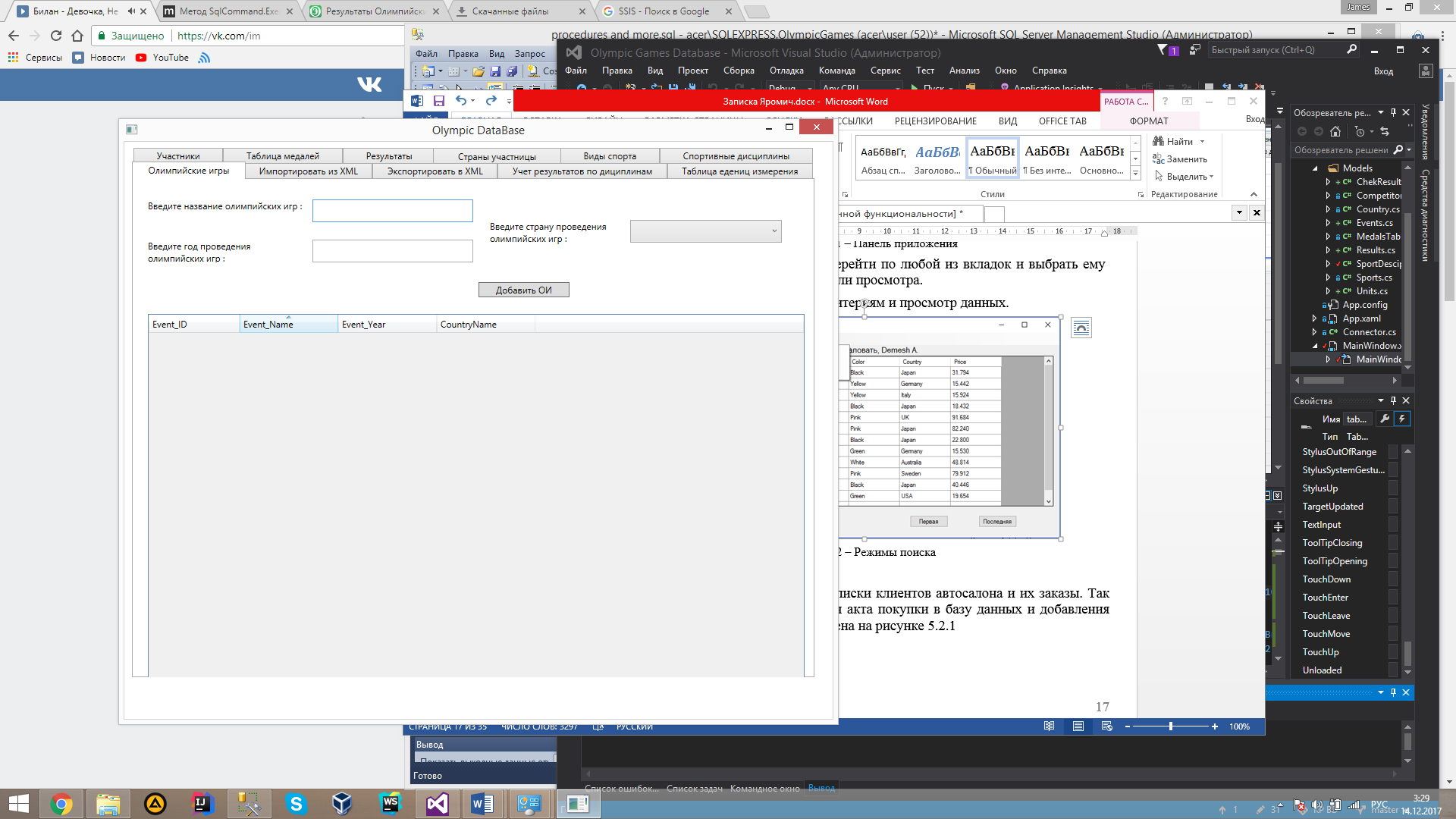


Рис. 5.1.2 – Использование WPF Datagrid для сортировка по полям

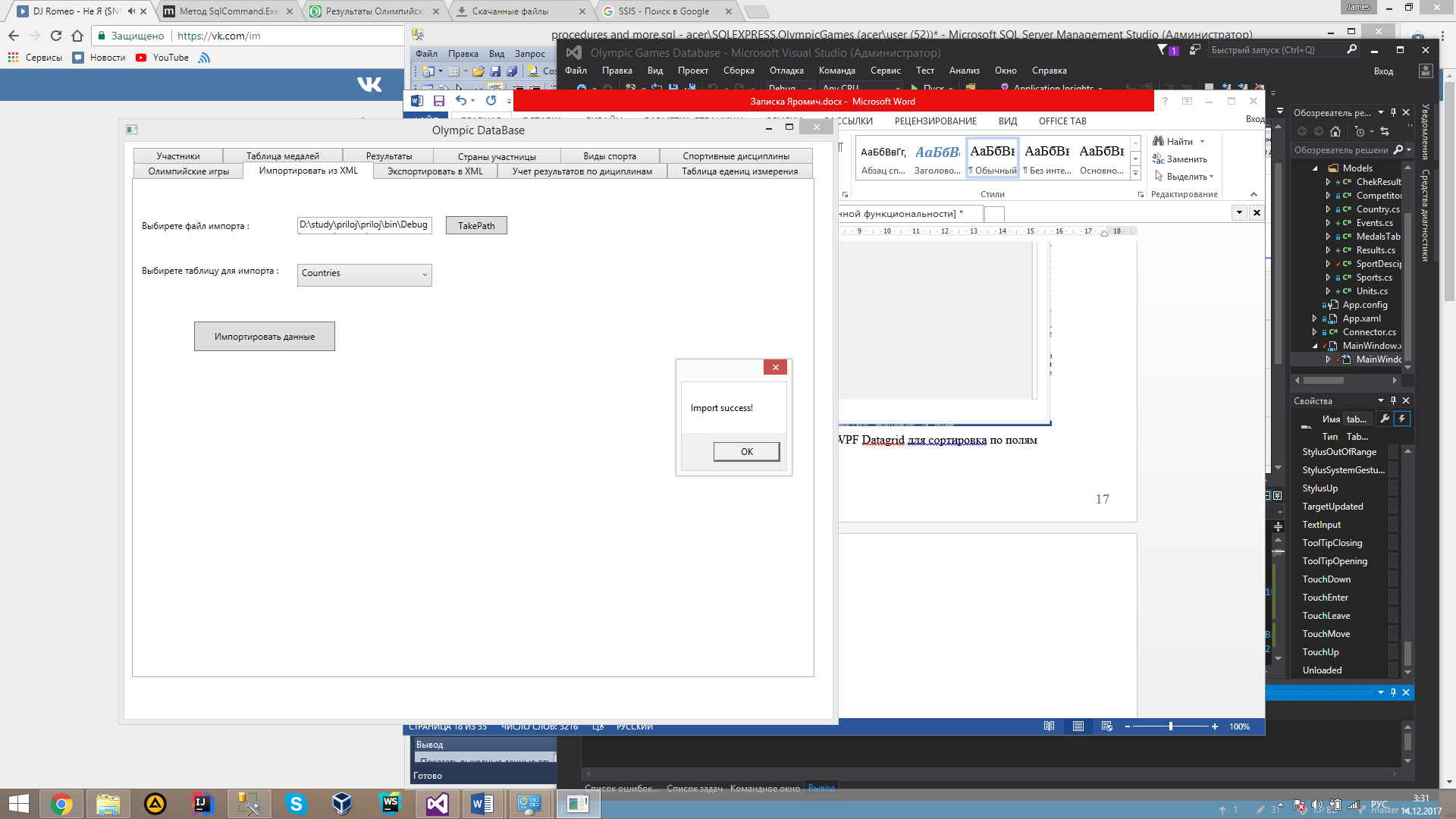


Рисунок 5.1.3 Импорт большого количество строк(100.000) в таблицу

Пользователь может экспортировать, импортировать данные как показано на рис.5.1.3. Все выполняется с помощью процедур и исключительно на них.

# Заключение

В заключение курсовой работы необходимо сделать вывод о результатах проделанной работы. В данном курсовом проекте была разработана база данных «Олимпийские игры». Программа полностью автоматизирует труд организаторов ОИ и организует надежное хранение и обработку данных.

В соответствии с полученным результатом работы базы данных можно сделать вывод, что разработанная БД работает верно, а требования технического задания выполнены в полном объеме.

Выполненные цели:

* Ввод-вывод информации о играх
* Учет текущих олимпийских игр
* Подсчет рейтинга стран на играх
* Оформление отчета о олимпийских играх за все года

При написании курсовой работы был использован API-интерфейс WPF, что позволило сделать приложение более красивым и упростило разработку в разы.

Также стоит отметить то, что при написании курсового проекта для достижения желаемых результатов, было изучено множество возможностей технологии .NET.

# Список использованных источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. [Шилдт](https://oz.by/people/more9048119.html), Г. Полный справочник по C# / Г. Шилдт. - «Вильямс», 2004. — 772 c.

# ПРИЛОЖЕНИЕ А

**SQL – скрипты созданных таблиц**

create procedure CreateTable

as

create table Units

(

UnitId int primary key identity(1,1),

UnitVerible nvarchar(50) unique

)

create table Gender

(

Gender\_Code int primary key identity(1,1),

Gender\_Name nvarchar(50) unique

)

CREATE TABLE Sports

(

Sport\_Code int primary key identity(1,1),

Sport\_Description nvarchar(50) unique

)

CREATE TABLE Countries

(

Country\_Code int primary key identity(1,1),

Country\_Names nvarchar(50) unique

)

create table SortType

(

SortTypeId int primary key identity(1,1),

SortTypeName nvarchar(4)

)

create table \_Event

(

Event\_ID int primary key identity(1,1),

Event\_Name nvarchar(50) unique,

Event\_Year date unique,

Country\_Code int not null FOREIGN KEY REFERENCES Countries(Country\_Code)

)

create table SportTypeDescription

(

SportTypeId int primary key identity(1,1),

Sport\_Code int not null FOREIGN KEY REFERENCES Sports(Sport\_Code),

SportTypeName nvarchar(50) unique,

UnitId int not null FOREIGN KEY REFERENCES Units(UnitId),

SortTypeId int not null FOREIGN KEY REFERENCES SortType(SortTypeId)

)

CREATE TABLE Competitors

(

Competitor\_ID int primary key identity(1,1),

Country\_Code int not null FOREIGN KEY REFERENCES Countries(Country\_Code),

Gender\_Code int not null FOREIGN KEY REFERENCES Gender(Gender\_Code),

Date\_of\_Birth date,

First\_Name nvarchar(50),

Last\_Name nvarchar(50)

)

CREATE TABLE Results

(

Results\_ID int primary key identity(1,1),

Competitor\_ID int not null FOREIGN KEY REFERENCES Competitors(Competitor\_ID),

Event\_ID int not null FOREIGN KEY REFERENCES \_Event(Event\_ID),

SportTypeId int not null FOREIGN KEY REFERENCES SportTypeDescription(SportTypeId),

Result nvarchar(50)

)

CREATE TABLE Medals\_Table

(

Event\_ID int not null FOREIGN KEY REFERENCES \_Event(Event\_ID),

Country\_Code int not null FOREIGN KEY REFERENCES Countries(Country\_Code),

Gold\_Medal\_Count int,

Silver\_Medal\_Count int,

Bronze\_Medal\_Count int,

Total\_Medal\_Count int

)

# ПРИЛОЖЕНИЕ Б

**Процедуры**

-- Экспорт в XML

IF OBJECT\_ID('exportCountriesToXML') IS NOT NULL

BEGIN

DROP PROC exportCountriesToXML

END

GO

CREATE PROC exportDataToXML @path nvarchar(256), @nametable nvarchar(50)

AS

begin

BEGIN TRAN

declare @sql nvarchar(500)=

'bcp "SELECT \* '+

'FROM '+ @nametable +

' FOR XML PATH(''Order''), ROOT(''Root'')" queryout "'+@path+'" -S acer\SQLEXPRESS -d OlympicGames -w -T ';

EXEC xp\_cmdshell @sql;

COMMIT;

end;

GO

--Импорт из XML

CREATE PROC insertCountryFromXML

@path nvarchar(256)

AS

begin

SET NOCOUNT ON

SET XACT\_ABORT ON

declare @count1 int=0;

declare @count2 int=0;

set @count1 = (select count(\*) from Countries)

BEGIN TRAN

declare @results table (x xml)

declare @sql nvarchar(300)=

'SELECT

CAST(REPLACE(CAST(x AS VARCHAR(MAX)), ''encoding="utf-16"'', ''encoding="utf-8"'') AS XML)

FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';

INSERT INTO @results EXEC (@sql)

declare @xml XML = (SELECT TOP 1 x from @results);

declare @ad nvarchar(50);

set @ad = ( SELECT

C3.value('Country\_Names[1]', 'NVARCHAR(100)') AS [CountryName]

FROM @xml.nodes('Root/Order') AS T3(C3) );

if @ad is null

begin

INSERT INTO [Countries]([Country\_Names])

SELECT

C3.value('Country\_Names[1]', 'NVARCHAR(100)') AS [CountryName]

FROM @xml.nodes('Root/Order') AS T3(C3)

end;

COMMIT;

set @count2 = (select count(\*) from Countries)

return @count2 - @count1;

end;

GO

--Добавление участника

create procedure AddCompetitor @Country\_Names nvarchar(50),

@Gender\_Name nvarchar(50),

@Date\_of\_Birth date,

@First\_Name nvarchar(50),

@Last\_Name nvarchar(50)

as begin

declare @Par nvarchar(50);

declare @SecondPar nvarchar(50);

declare @GendCode int;

declare @CountryCode int;

set @GendCode = ( Select Gender.Gender\_Code

from Gender

where Gender\_Name = @Gender\_Name

);

set @CountryCode = ( Select Country\_Code

from Countries

where Country\_Names = @Country\_Names

);

begin

insert into Competitors(Country\_Code,Gender\_Code,Date\_of\_Birth,First\_Name,Last\_Name)

values (@CountryCode,@GendCode,@Date\_of\_Birth,@First\_Name,@Last\_Name);

end

end

go

--Выборка из таблицы результатов

create procedure SelectResults

as begin

select t1.Results\_ID, t2.Event\_Name,t3.First\_Name, t3.Last\_Name, t4.SportTypeName, t1.Result from Results as t1

inner join \_Event as t2 on (t1.Event\_ID = t2.Event\_ID)

inner join Competitors as t3 on (t1.Competitor\_ID = t3.Competitor\_ID)

inner join SportTypeDescription as t4 on (t1.SportTypeId = t4.SportTypeId);

end

go

--Добавление в таблицу результатов

create procedure AddResult @First\_Name nvarchar(50),

@Last\_Name nvarchar(50),

@Event\_Name nvarchar(50),

@SportTypeName nvarchar(50),

@Result nvarchar(50)

as begin

declare @CompId int;

set @CompId = ( select Competitor\_ID from Competitors where First\_Name = @First\_Name and Last\_Name = @Last\_Name );

declare @EvenId int;

set @EvenId = ( select Event\_ID from \_Event where Event\_Name = @Event\_Name);

declare @SportTypeId int;

set @SportTypeId = ( select SportTypeId from SportTypeDescription where SportTypeName = @SportTypeName);

insert into Results(Competitor\_ID,Event\_ID,SportTypeId,Result)

values (@CompId,@EvenId,@SportTypeId,@Result);

end

go

# ПРИЛОЖЕНИЕ В

**Наиболее важные функции**

private void Import()

{

string param = string.Empty;

string par = string.Empty;

this.Dispatcher.BeginInvoke(System.Windows.Threading.DispatcherPriority.Normal, (ThreadStart)delegate ()

{

param = TableBoxExp.Text;

par = PathImport.Text;

using (SqlConnection cn = Connector.GetConnection())

{

cn.Open();

switch (param)

{

case "Countries":

{

SqlCommand cmd = new SqlCommand("insertCountryFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

GetCountries();

}

break;

case "Competitors":

{

SqlCommand cmd = new SqlCommand("insertCompetitorsFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

}

break;

case "Sports":

{

SqlCommand cmd = new SqlCommand("insertSportsFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

SqlParameter Country = new SqlParameter();

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

} break;

case "Gender":

{

SqlCommand cmd = new SqlCommand("insertGenderFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

}break;

case "SportTypeDescription":

{

SqlCommand cmd = new SqlCommand("insertSportTypeDescriptionFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

}

break;

case "Units":

{

SqlCommand cmd = new SqlCommand("insertUnitsFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

}

break;

case "\_Event":

{

SqlCommand cmd = new SqlCommand("insertEventFromXML", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@path", par);

SqlDataReader data = cmd.ExecuteReader();

}

break;

}

MessageBox.Show("Import success!");

cn.Close();

}

});

}

private void ImportData\_Click(object sender, RoutedEventArgs e)

{

Thread thr = new Thread(new ThreadStart(Import));

thr.Start();

}

private void AddCompetitor\_Click(object sender, RoutedEventArgs e)

{

this.Competitors.Items.Clear();

using (SqlConnection cn = Connector.GetConnection())

{

cn.Open();

SqlCommand cmd = new SqlCommand("AddCompetitor", cn);

cmd.CommandType = System.Data.CommandType.StoredProcedure;

SqlParameter Country = new SqlParameter();

cmd.Parameters.AddWithValue("@Country\_Names", Countryiess.SelectedItem.ToString());

cmd.Parameters.AddWithValue("@Gender\_Name", Genders.SelectedItem.ToString());

cmd.Parameters.AddWithValue("@First\_Name", FirstNameBox.Text);

cmd.Parameters.AddWithValue("@Last\_Name", LastNameBox.Text);

cmd.Parameters.AddWithValue("@Date\_Of\_Birth", Birthd.Text.ToString());

SqlDataReader data = cmd.ExecuteReader();

while (data.Read())

{

Competitor ap = new Competitor(Convert.ToInt32(data[0].ToString()), data[1].ToString(),

data[2].ToString(), data[3].ToString(),

data[4].ToString(), data[5].ToString());

this.Competitors.Items.Add(ap);

}

cn.Close();

}

GetCompetitors();

}

# ПРИЛОЖЕНИЕ Г.

Генератор данных

var xmldoc = new XDocument(

new XElement("Root"));

Random ran = new Random();

for (int i = 0; i < 50000; i++)

{

string Name = "";

string First\_Name = GetPass(ran.Next(3, 10));

string Last\_Name = GetPass(ran.Next(3, 10));

int year = ran.Next(1970, 1999);

int month = ran.Next(1, 12);

int day = ran.Next(1, 28);

string date = String.Format("{0}-{1}-{2}", year, month, day);

Name += i;

xmldoc.Root.Add(new XElement("Order", new XElement("First\_Name", First\_Name),

new XElement("Last\_Name", Last\_Name),

new XElement("Country\_Code", ran.Next(1, 100)),

new XElement("Date\_Of\_Birth", date),

new XElement("Gender\_Code", ran.Next(1, 2))));

}

for (int i = 0; i < 200; i++)

{

string Country\_Names = GetPass(ran.Next(4, 10));

xmldoc.Root.Add(new XElement("Order", new XElement("Country\_Names", Country\_Names)));

}

for (int i = 0; i < 50000; i++)

{ string result = String.Format("{0}.{1}", ran.Next(7, 100), ran.Next(1, 980));

xmldoc.Root.Add(new XElement("Order", new XElement("Competitor\_ID", ran.Next(1, 50000)),

new XElement("Event-ID", ran.Next(1, 28)),

new XElement("SportTypeID", ran.Next(1, 30)),

new XElement("Result", result)));

}

xmldoc.Save("results.xml");

}

public static string GetPass(int x)

{

string pass = "";

var r = new Random();

while (pass.Length < x)

{

Char c = (char)r.Next(33, 125);

if (Char.IsLetter(c))

pass += c;

}

return pass;

}