**Экспорт и импорт данных**

**Экспорт** – сохранение данных БД во внешний файл

**Импорт** – загрузка данных в БД из внешнего файла

БД(ПКМ) –> Задачи -> Импорт/Экспорт данных

При экспорте сервер источник, при импорте сервер приёмник

SQL Server Native Client / Microsoft OLE DB Provider for SQL Server

**Стандартные типы файлов экспорта/импорта:**

* **Excel (xls/xlsx)** – Данные каждой таблицы на отдельном листе

XLS 97-2003

XLSX 2007-… - Если файл в этом файле, то надо пере сохранить как XLS

После импорта может потребоваться изменить название столбцов и таблиц, типы данных и обязательность, добавить PK FK

* **Flat File (csv/tsv/txt)** – Таблица хранится в текстовом формате, может быть открыт в табличном редакторе. Данные разделяются [, ; пробел \t (табуляция) |], перенос строки Enter \n

При ошибке загрузки строковых данных нужно поменять тип **DT\_STR -> DT\_TEXT**

Смена кодировки в Notepad++ или в блокноте -> Convert to ANSI

Можно **пересохранить csv как xls**

* **XML** – расширяемый режим разметки, можно создавать свои теги и атрибуты

**Особенности XML:**

* HTML / HAML
* Есть один корневой тег ROOT
* Каждый тег должен быть закрыт <тэг1>…</тэг1> <тэг2/>
* В открывающемся теге могут быть указаны атрибуты

<тэг1 атр1=”знач” атр2=”знач2”>

* Названия тегов и атрибутов не могут содержать пробелы и спец. Символы
* Между тегами могут быть другие значения или другие теги
* Ошибки можно найти, открыв файл в браузере
* Можно открыть в Excel и пересохранить

Пример экспорта XML файла:

SELECT \*

FROM Таблица

FOR XML PATH('Название узла/строки'), ROOT('Название корневого узла/таблицы')

SELECT \*

FROM Game

FOR XML PATH('Game'), ROOT('Games')

Пример импорта с помощью SQL:

1. Объявление переменной

DECLARE @xml XML = '...';

1. Общая форма

<Products>

<Category name="Электроника">

<Product>

SELECT

Объект.value('Имя тега столбца[1]', 'тип данных') AS Столбец,

...

Объект.value('../@атрибут родительского тега', 'тип данных') AS Столбец,

FROM @xml.nodes('/Тег корневой/Тег родительский/Тег объекта') AS Таблица(Объект);

SELECT

product.value('ProductId[1]', 'INT') as ProductId,

product.value('Name[1]', 'NVARCHAR(100)') as Name,

product.value('Price[1]', 'DECIMAL(10, 2)') as Price,

product.value('@attr', 'INT') as attr,

product.value('../@name', 'NVARCHAR(100)') as Category

FROM @xml.nodes('/Products/Category/Product') AS t(product)

SELECT

ДочернийОбъект.value('Имя тега столбца[1]', 'тип данных') AS Столбец,

...

РодительскийОбъект.value('@атрибут родительского тега', 'тип данных') AS Столбец,

FROM @xml.nodes('/Тег корневой/Тег родительский/Тег объекта') AS Таблица1(РодительскийОбъект)

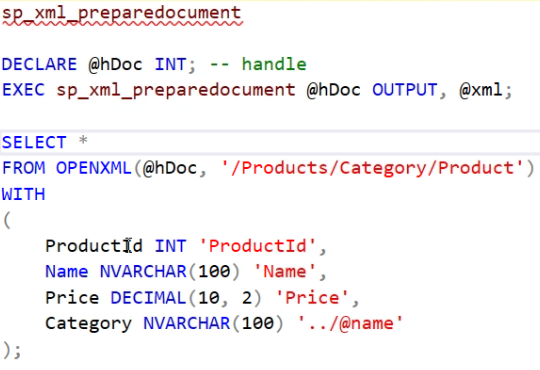
CROSS APPLY Таблица1.РодительскийОбъект.nodes('Тег дочерний') AS Таблица2(ДочернийОбъект)

Разное название родительских тегов:

Вместо названия тега используется local-name(..)

Путь указывается вместо родительского тега “\*” /Тег корневой/\*/ Тег дочерний

Для считывания можно использовать sp\_xml\_preparedocument



* **Json** (JavaScript Object Notation)

**Особенности JSON:**

* [] Массив
* {} Объект
* “атр1”: знач1 / “атр2”: “знач2”
* Значение может быть и массив и другой объект

**Пример экспорта:**

SELECT \*

FROM Game

FOR JSON PATH, INCLUDE\_NULL\_VALUES

**Импорт JSON:**

Используется OPENJSON

Обращение к значению '$.атрибут'

SELECT \*

FROM OPENJSON(@json)

WITH

(

ProductId INT '$.ProductId',

Name NVARCHAR(100) '$.Name',

Price DECIMAL(10, 2) '$.Price',

Category NVARCHAR(100) '$.Category'

);

Типы данных лучше настраивать в проекте