МИНОБРНАУКИ РОССИИ

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Южный федеральный университет»**

**Институт математики, механики и компьютерных наук
им. И.И. Воровича**

Гончаров Виктор Владимирович

Применение XML технологий на MS SQL для сопровождения корпоративных приложений

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению 02.03.02 – Фундаментальная информатика
и информационные технологии**

**Научный руководитель –
доцент, к.т.н. Литвиненко Александр Николаевич**

Ростов-на-Дону – 2018

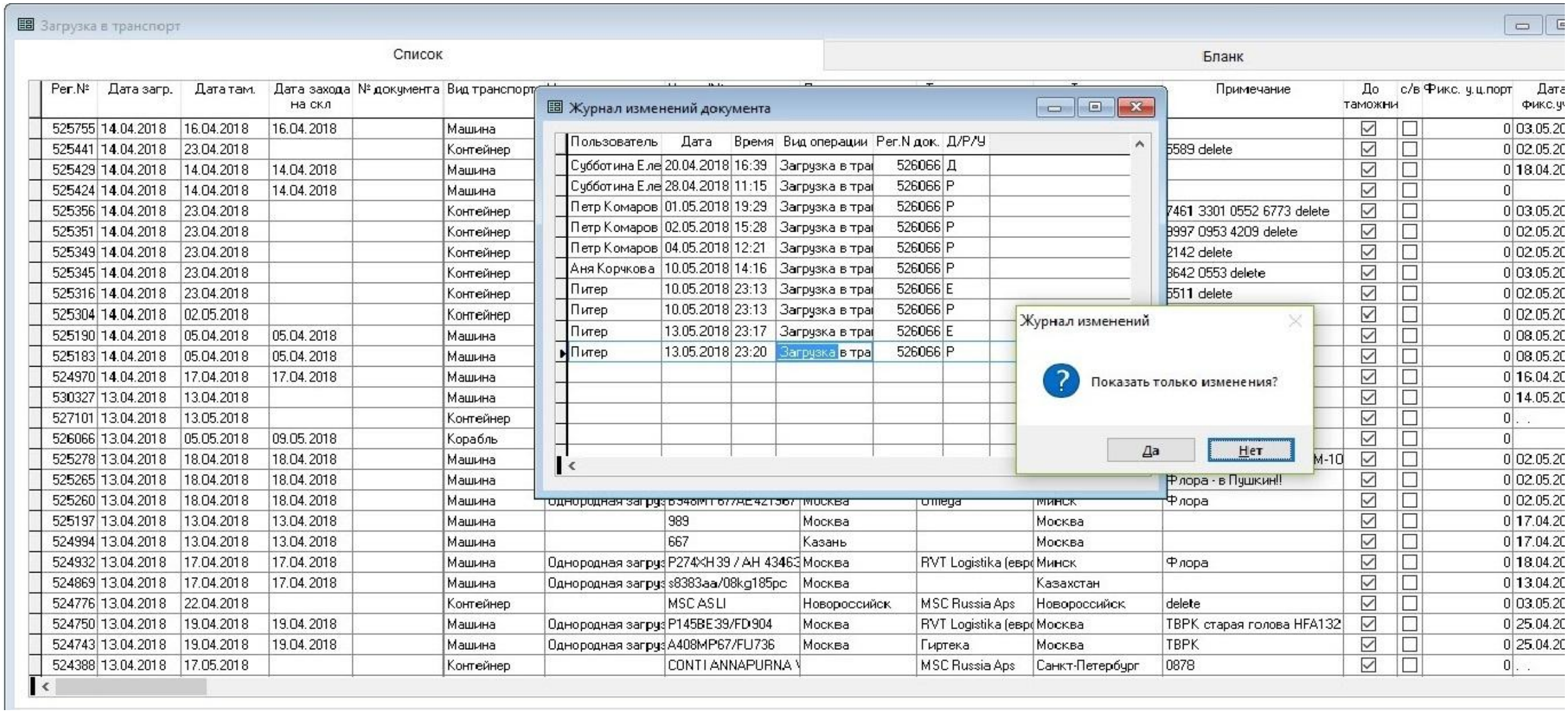
Постановка задачи

Одной из задач администрирования корпоративных приложениях является ведение журнала изменений.

Он необходим для того, чтобы понять кто, когда и как вносил изменения в документы. Это нужно для разбора конфликтных ситуаций и решения различных административных задач.

В работе решается ряд технических проблем, связанных с реализацией и ведением такого журнала.

Пример использования журнала изменений в корпоративном приложении



Цели и задачи

Целью данной работы является реализация методов обработки XML полей таблицы состояний документов.

Для достижения этой цели необходимо решить следующие задачи:

- Реализация процедуры преобразования табличного представления документов в XML представление.
- Создание процедуры определения различий между двумя XML представлениями (DIFF).
- Реализация процедуры обратного преобразования, позволяющего по предыдущему состоянию и дельте построить текущее состояние документа (PATCH).

Структура таблицы для журнала в базе данных

Column Name	Data Type	Allow Nulls
LOGU_id	int	☐
shluz_id	int	☑
dt	datetime	☐
docname	char(2)	☐
doc_id	int	☐
rwd	char(1)	☐
dxml	xml	☑
dxml3	xml	☑
dxml0	xml	☑

Сравнение с существующими решениями для задач

DIFF и PATCH

- DeltaXMLCore
- Microsoft Diff and Patch Tool
- WinMerge
- JExamXML

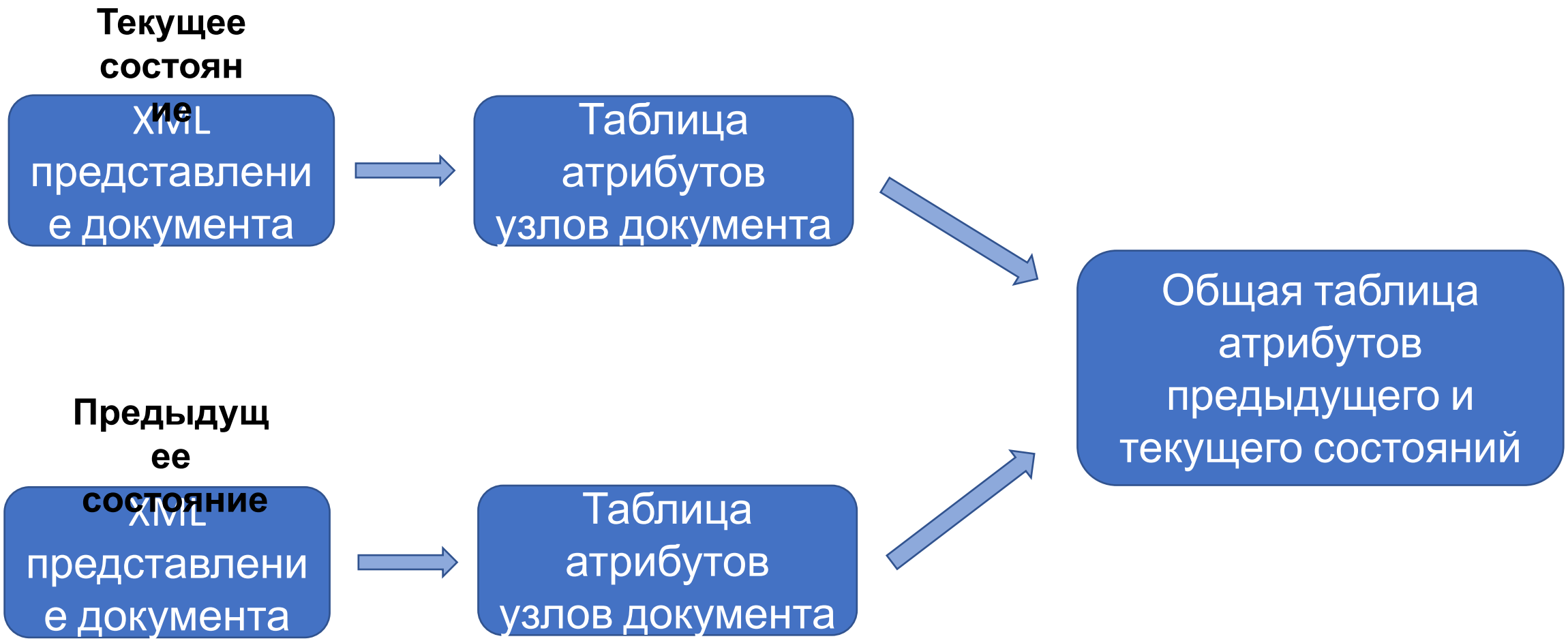
Данные модули не подходят для решения наших задач, так как нужно сравнивать документы, учитывая ключевые атрибуты в узлах.

Преобразование реляционного представления в XML

- Формирование корневого узла, который соответствует главной таблице
- Создание вложенных узлов для дочерних таблиц документа

```
SELECT *
    ,( SELECT * FROM rs WHERE rs.norg_id = slorg.norg_id FOR XML RAW('rs'), type  )
    ,( SELECT * FROM sklorg WHERE sklorg.norg_id = slorg.norg_id FOR XML RAW('sklorg'), type  )
    ...
    ,( SELECT * FROM pltovo WHERE slorg.norg_id = pltovo.norg_id FOR XML RAW('pltovo'), type  )
FROM slorg
WHERE norg_id = @Obj_id
FOR xml raw ('slorg'), type
```

Нахождение дельты. Схема процесса формирования общей таблицы атрибутов



Переход от XML к табличному представлению

- Документ разбивается на логические копии с помощью метода nodes
- Последовательно извлекаются имя и значение каждого атрибута, а также имя и ID узла, в котором он находится.

```
SELECT    T.N.value('local-name(.)', 'nvarchar(100)') as AtrName
         ,T.N.value('.', 'nvarchar(100)') as Value
         ,T.N.value('local-name(..)', 'nvarchar(100)') as NodeName
         ,coalesce( T.N.value('../@*[1]', 'nvarchar(100)'),'' ) as d_id
FROM @xml2.nodes('//@*') AS T(N)
)
```

Результат преобразования XML представления к табличному

```
<uz UZ_id="146837" marka="0"  >
    <uz1 UZ1_id="2808225" uz_id="146837" uzp_id="132464" attrizm="222"/>
    <uz1 UZ1_id="2808227" uz_id="146837" uzp_id="132464" />
</uz>
```

	AtrName	Value	NodeName	d_id
1	UZ_id	146837	uz	146837
2	marka	0	uz	146837
3	dat	2018-04-20T00:00:00	uz	146837
4	dobatr	zndobatr	uz	146837
5	UZ1_id	2808225	uz1	2808225
6	dobatr1	zndobatr1	uz1	2808225
7	attrizm	333	uz1	2808225
8	uz_id	146837	uz1	2808225
9	UZ1_id	2808226	uz1	2808226
10	uz_id	146837	uz1	2808226
11	uzp_id	132464	uz1	2808226

Формирование общей таблицы атрибутов узлов двух состояний

	NodeName0	AtrName0	pk1	pk2	ValueNew	Valueold
1	uz	dat	146837	NULL	2018-04-20T00:00:00	NULL
2	uz	dobatr	146837	NULL	zndobatr	NULL
3	uz1	uzp_id	NULL	2808225	NULL	132464
4	uz1	UZ1_id	NULL	2808227	NULL	2808227
5	uz1	uz_id	NULL	2808227	NULL	146837
6	uz1	uzp_id	NULL	2808227	NULL	132464
7	uz1	dobatr1	2808225	NULL	zndobatr1	NULL
8	uz1	attrizm	2808225	2808225	333	222
9	uz1	UZ1_id	2808226	NULL	2808226	NULL
10	uz1	uz_id	2808226	NULL	146837	NULL
11	uz1	uzp_id	2808226	NULL	132464	NULL

Формирование результирующей дельты из полученных таблиц

1. Определение статуса изменения узла
2. Добавление элементов в узел, учитывая статус изменений:

```
SELECT ' '+RTRIM(atrname0)+'="'+RTRIM(coalesce(Valueold,'NULL'))
    +':'+RTRIM(coalesce(Valuenew,'NULL'))+'"' AS 'data()'
FROM #d1
WHERE (coalesce(pk1,pk2,'') = #d2.pk1 OR coalesce(pk1,pk2,'') = #d2.pk2)
    AND #d1.NodeName0 = #d2.NodeName0
    AND #d1.atrname0 != #d2.atr_1
for xml path('')
```

3. Объединение сформированных узлов в результирующую дельту.

Обратное преобразование

Нахождение текущего состояние документа по предыдущему состоянию и их дельте.

Реализуется следующим образом:

- Последовательная обработка каждого узла дельты.
- Модификация предыдущего состояния по полученным данным.

Результаты работы

- Реализована хранимая процедура для преобразования реляционного представления документов в XML представление.
- Создана процедура определения различий между двумя XML представлениями.
- Реализована процедура обратного преобразования, позволяющего по предыдущему состоянию и дельте построить текущее состояние документа.