МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» марта 2016 г.

ГРУППОВАЯ ОБРАБОТКА ДАННЫХ В СИСТЕМЕ DEDUCTOR STUDIO ПРИ РАБОТЕ С ХРАНИЛИЩЕМ ДАННЫХ

Методические указания к лабораторной работе № 8
для студентов направления 09.03.02 и 09.03.03

Курск 2016

УДК 004
Составитель А.В. Ткаченко

Рецензент
Кандидат технических наук, доцент С.Ю. Сазонов

Групповая обработка данных в системе Deductor Studio при работе с хранилищем данных: методические указания к лабораторной работе № 8 по дисциплине «Технологии обработки информации» / Юго-Зап. гос. ун-т; сост. А.В. Ткаченко. Курск, 2016. 7 с. Библиогр.: с. 7.

Приводится описание технологии групповой обработки данных в системе Deductor Studio при работе с хранилищем данных.

Методические указания соответствуют требованиям утвержденной рабочей программы дисциплины.

Предназначены для студентов, обучающихся по направлениям: 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика».

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы: Освоить технологию групповой обработки данных в системе **Deductor Studio** при работе с хранилищем данных.

Узел **Групповая обработка** работает похожим на **Скрипт** образом. Основным отличием от него является то, что входной набор делится на части по указанным группам, и затем каждая группа отдельно «прогоняется» через копию цепочки узлов обработки.

Если аналогом скрипта является *процедура* в языке программирования, то аналогом групповой обработки - *цикл.* Групповая обработка позволяет создавать очень гибкие сценарии, особенно она незаменима в тех случаях, когда нужно обрабатывать отдельные «пачки» данных внутри одного набора в зависимости от статистических характеристик каждой такой «пачки» (сумма, среднее, количество записей и т.д.).

Рассмотрим групповую обработку на конкретном примере. Импортируем в **Deductor** текстовый файл **Trade.txt** (по умолчанию он расположен в каталоге /Samples). Фрагмент набора данных приведен ниже в таблице.

Дата (Год+Месяц)	Количество
2010-M01	462 523,419
2011-M02	633 203,196
2012-M03	660159,299
2013-M04	617 455,341
2014-M05	597 354,479

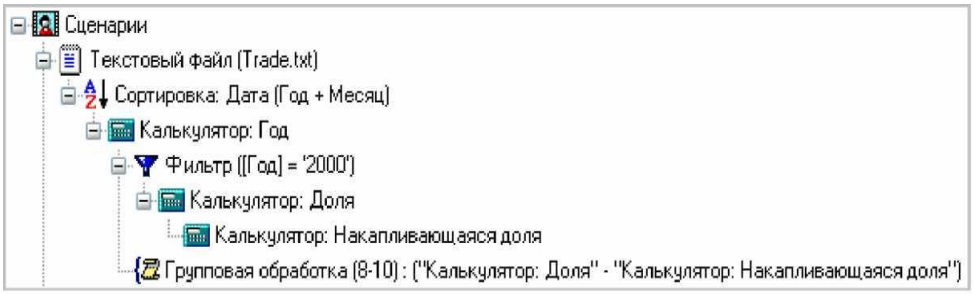
Отсортируем его по возрастанию по полю Дата (Год + Месяц). Далее из этого поля узлом **Калькулятор** выделим год, создав новое поле с функцией SUBSTR(COL1;1;4).

Пусть перед нами стоит задача: рассчитать для каждого месяца каждого года (т.е., по сути, строки набора данных) долю и долю с накоплением от годовой суммы в пределах одного года.

Ситуация характеризуется тем, что у нас не один год, а несколько: с 2010 по 2014.

Воспользуемся **Групповой обработкой.** Для наглядности сначала проделаем все необходимые действия над одной группой,

скажем, 2010 год, а затем «распространим» эти действия на весь исходный набор данных.



Сначала мы выделили эту группу фильтром и последовательно добавили два поля двумя калькуляторами: Доля (PART):

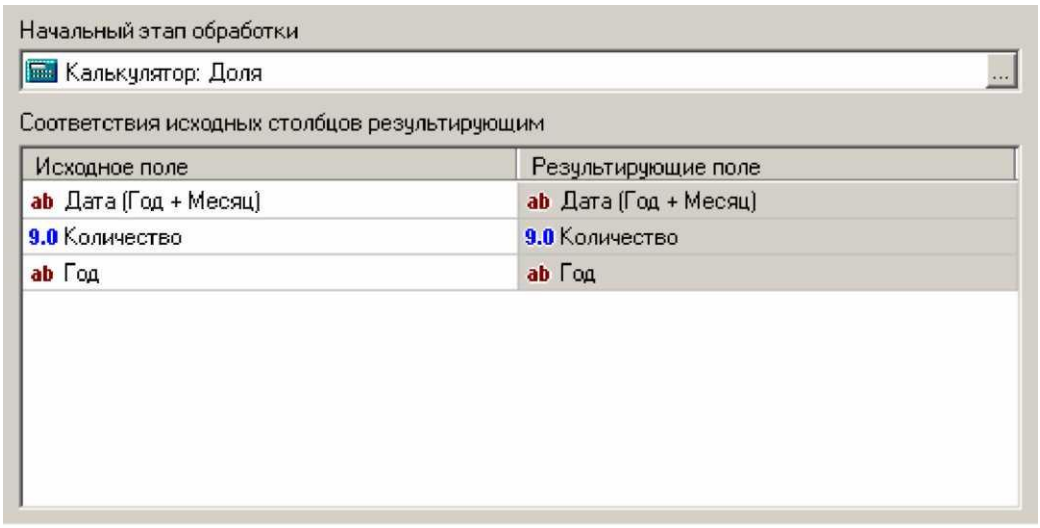
ROUND(COL2/Stat("COL2";"SUM")*100;2),

и Накапливающаяся доля (CUMPART):

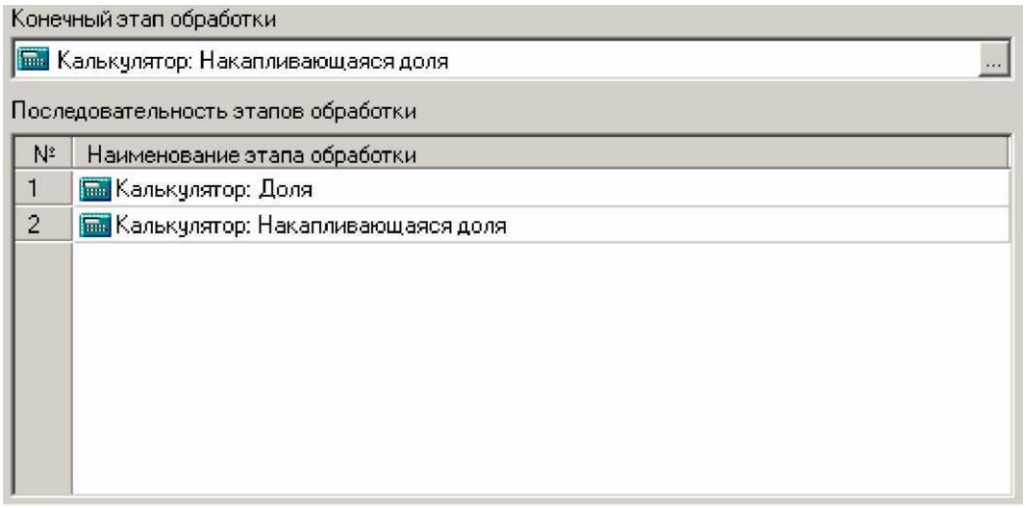
CumulativeSum("PART").

Далее добавим к исходному набору данных узел **Групповая обработка.** На первом шаге нужно указать поля для определения групп при обработке данных. В нашем случае это поле Год.

На следующей вкладке укажем начальный этап обработки - узел с меткой Калькулятор: Доля.



Конечным узлом будет Калькулятор:



В результате групповой обработки получим следующий набор данных (на рисунке изображен фрагмент набора).

Год	**Дата (Год + Месяц)**	**Доля**	**Накапливающаяся доля**	**Количество**
2010	2010-**M** 01	4**,16**	4	462523,418
2011	2011-**M** 02	5,7	10	633203,186
2012	2012-**M** 03	5,34	16	660158,288
2013	2013-**M** 04	5,56	21	617455,3417
2000	2000-**M** 05	5,33	27	537354,4784
12000	2000-**M** 06	7,14	34	733517,4512
2000	2000-**M** 07	3,15	43	1015844,2862

2000	2000-**M** 08	10,34	53	1143052,2523
12000	2000-**M** 03	10,41	64	1156623,1715
2000	2000-**M** 10	**11**,3	75	1255021,3423
2000	2000-**M 11**	**12**,7	83	1410114,5606
2000	2000-**M** 12	12,22	100	1357230,3388
2001	2001-**M** 01	5,16	5	1003317,7317
J2001	2001-**M** 02	5,64	**11**	1037048,6263
12001	2001-**M** 03	7,71	13	1433877,3427
12001	2001-**M** 04	7,76	26	1507686,4482
12001	2001-**M** 05	7,32	34	1520761,5588
12001	2001-мое	3,25	42	1602674,5245
12001	2001-**M** 07	3,67	51	1685883,1625
12001	2001-**M** 03	3,77	61	1888255,345
12001	2001-**M** 03	3,33	70	1716804,1633
12001	2001-**M** 10	10,65	80	2068772,3382
12001	2001-**M 11**	10,37	81	2016227,4267
12001	2001-**M** 12	3,35	100	1817580,4566
12002	2002-**M** 01	6,13	6	1433788,5082

Обратите внимание - накапливающаяся доля доходит до 100% в каждом году, и «сбрасывается» с началом нового года. Таким образом, мы получили желаемый результат. Без групповой обработки получить это было бы гораздо сложнее.

На первой вкладке мастера настройки узла были доступны три опции. Разберем их детальнее.

Флаг **Переобучать модель всегда и для каждой группы** актуален, когда в цепочке узлов, на которые ссылается групповая обработка, имеются какие-либо модели - линейная регрессия, нейронная сеть и так далее. Поэтому в случае простых действий - **Калькулятор, Фильтр, Замена данных, Сортировка** и другие - на данный флаг не нужно обращать внимания.

Флаг **Пропускать группы с ошибками** исключит из результирующего набора группы, при «прогоне» которых через цепочку узлов возникла ошибка. В подавляющем большинстве случаев это бывает также при наличии в цепочке узлов каких-либо моделей, поэтому при простых действиях флаг ставить не нужно.

Флаг **Использовать кэш для результата** определяет один из двух вариантов функционирования узла: «без использования кэширования» и «с использованием кэширования».

Определение

Кэш - это подборка данных, дублирующих оригинальные значения, сохранённые где-то или вычисленные ранее, когда оригинальные данные труднодоступны из-за большого времени доступа или для вычисления. Многие программы записывают куда-либо промежуточные или вспомогательные результаты работы, чтобы не вычислять их каждый раз, когда они понадобятся. Это ускоряет работу, но требует дополнительной памяти (оперативной или дисковой).

Кэш требуется для экономии памяти. Это необходимо, когда групп обработки много и каждая группа требует больших вычислительных затрат. Большие вычислительные затраты, как правило, возникают при переобучении моделей - пересчете коэффициентов регрессии, подборе весов нейронной сети и так далее. Поэтому здесь совет следующий. Когда групп немного и в цепочке узлов «прогона» групп нет моделей, то кэш не нужен. В иных случаях лучше поставить флаг «с КЭШем».

Практическая работа:

Повторите в Deductor пример с групповой обработкой.

Вопросы для самоконтроля:

1. Для чего предназначен узел **Групповая обработка?**
2. В чем принципиальное отличие узла **Скрипт** от **Групповая обработка?**
3. Приведите примеры, когда может потребоваться **Групповая обработка.**
4. В каких случаях нужно включать флаг **Использовать кэш для результата?**

Библиографический список

1. Deductor Studio [Электронный ресурс]: www.basegroup.ru/download/deductor/.
2. Решения по построению хранилищ данных [Электронный ресурс]: http://ibarus.ru/solutions/dwh/?
3. Основные обработчики в Deductor Studio [Электронный ресурс]: http://deductor.org/Deductor_help_manual/deductor-help.html.