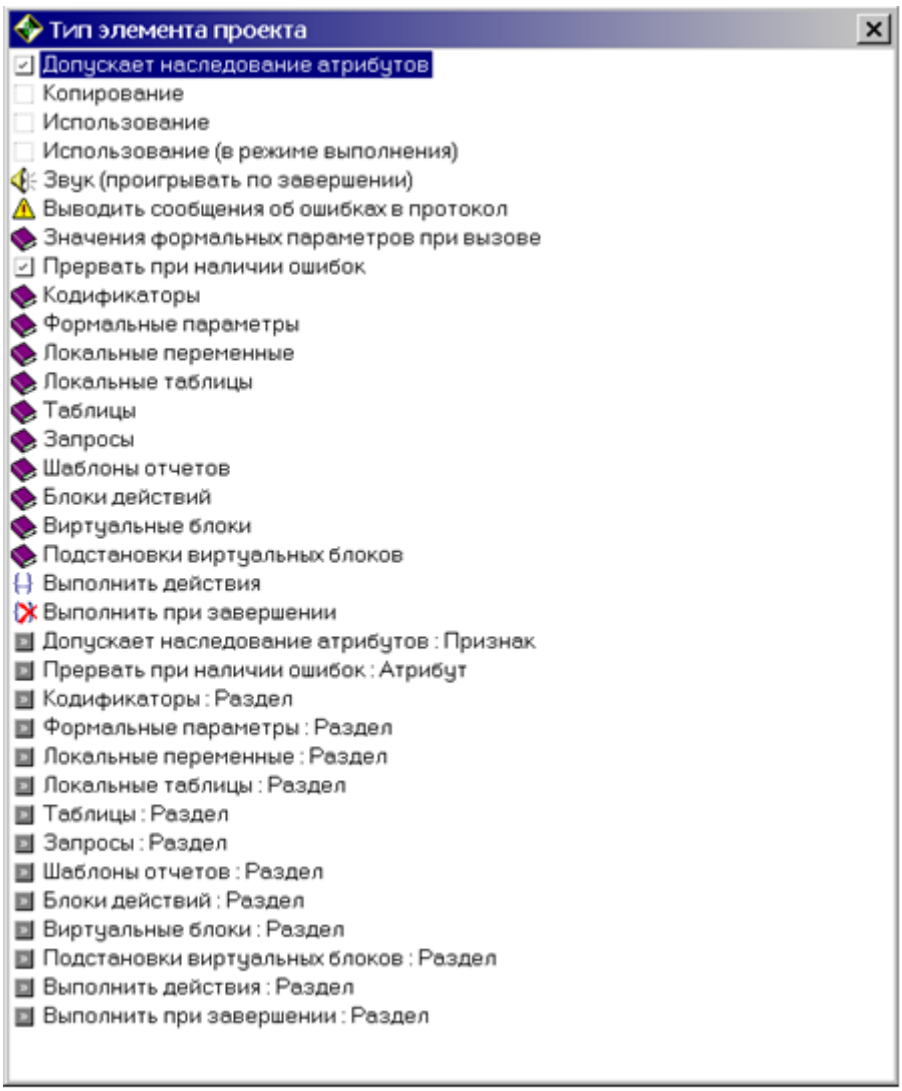
Структура процедуры (функции) на языке скриптов

Для описания процедуры (функции) на языке скриптов доступны следующие атрибуты:



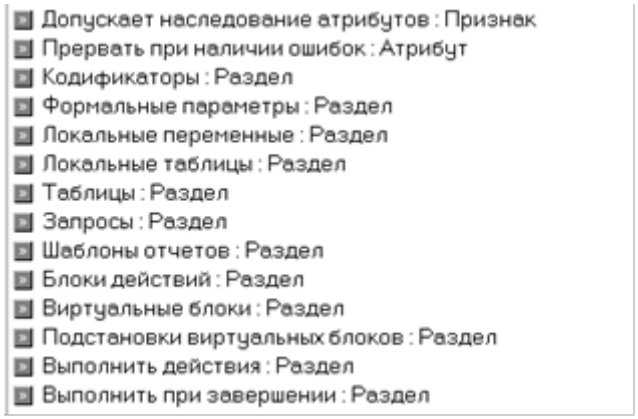
Атрибуты в начале списка являются общими для всех процедур, и предназначение этих атрибутов приведено ранее.

Рассмотрим подробнее остальные атрибуты.

- **Прервать при наличии ошибок** – при наличии этого признака главная процедура прерывается, если произойдет ошибка выполнения в запущенной из нее другой процедуре (функции, выражении). Если выполнялся основной блок действий, то гарантировано выполнится блок *‘Выполнить при завершении’*. Если же ошибка произошла в блоке ‘*Выполнить при завершении’*, то и он будет прерван. Отсутствие этого признака означает, что сообщение об ошибке записывается в протокол, и выполнение главной процедуры не прерывается.
- **Кодификаторы** - Здесь описываются используемые внутри процедуры кодификаторы. Кодификатор – это список именованных проектных элементов. Обычно кодификатор применяется для выбора пользователем одной альтернативы из списка возможных вариантов.
- **Локальные таблицы** – Здесь описываются используемые внутри процедуры локальные таблицы. Под локальной таблицей можно понимать двумерную матрицу из строк и столбцов с описанием правил сортировки и группировки строк.
- **Таблицы** - Здесь описываются используемые внутри процедуры таблицы сервера базы данных. Описанные здесь таблицы могут быть использованы в SQL конструкциях (операторах) процедуры.
- **Запросы** – Здесь описываются используемые внутри процедуры SQL запросы к серверу базы данных. Запросы могут быть описаны либо целиком, либо сюда могут быть вынесены их общие части (поскольку для SQL запросов поддерживается механизм наследования).
- **Шаблоны отчетов** – Здесь описываются шаблоны для процедур-отчетов на языке скриптов.
- **Блоки действий** – Здесь описываются блоки команд (операторов), вынесенные из основного блока процедуры. Блоки, описанные в этом разделе, в отличии от блоков основного тела процедуры, не вызываются автоматически, а требуют явного вызова.
- **Виртуальные блоки** – Используются при наследовании процедур с перенесением части логики в процедуру-наследник. Расположенные в этом разделе блоки не имеют содержимого. В теле процедуры содержатся ссылки на виртуальные блоки, но свое содержимое виртуальные блоки получают в разделе *‘Подстановки виртуальных блоков’* в процедуре-наследнике или базовой процедуре.
- **Подстановки виртуальных блоков** – В этом разделе виртуальные блоки получают свое содержимое в виде последовательности команд (операторов).
- **Выполнить действия -** Основной блок команд (операторов), автоматически выполняемый при запуске процедуры.
- **Выполнить при завершении -** Блок команд (операторов), автоматически выполняемый при завершении процедуры, т.е. после выполнения основного блока. Обычно в блоке ‘*Выполнить при завершении’* содержатся операторы для закрытия открытых в основном блоке ресурсов (файлы, выборки по файлам, basic-строкам и т.д.).

Процедуры закрытия ресурсов гарантировано выполнятся, даже если основной блок процедуры был прерван вследствие ошибки или по воле пользователя.

- **Наследуемые атрибуты** – задают наличие описанных выше разделов и признаков в процедуре-наследнике.

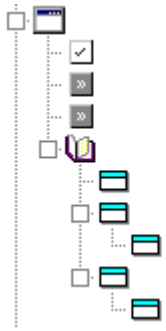


Чтобы процедура-наследник, в свою очередь, могла управлять доступом своих наследников к вышеуказанным разделам и признакам - в процедуре-родителе требуется установить признак ‘Допускает наследование атрибутов’. Наличие этого признака у процедуры-наследника определяется выбором этого признака в качестве наследуемого атрибута.

Локальные переменные

В этом разделе описываются локальные переменные процедуры или функции.

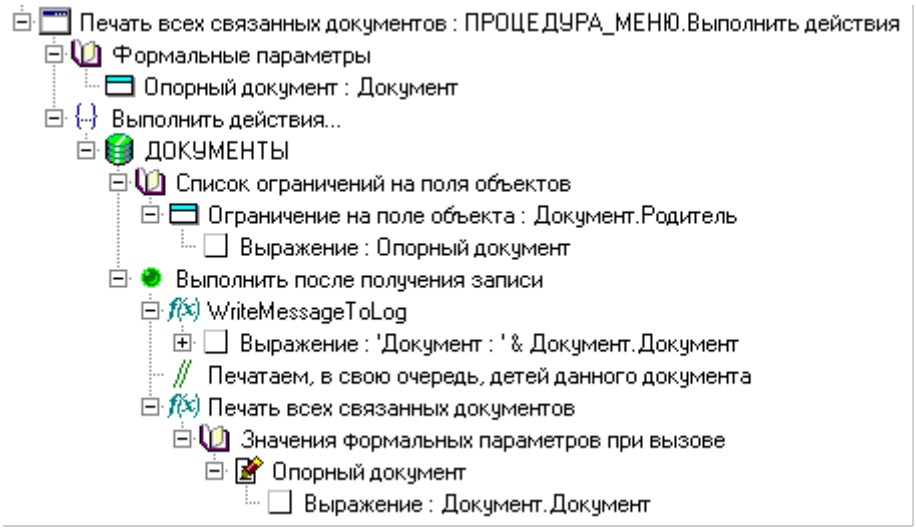
При описании локальной переменной возможно указать выражение для получения начального значения:



В описании процедуры указаны три локальные переменные. Для переменной ‘Сумма скидки’ задано начальное значение в виде константы. Начальное значение для переменной ‘Торговое подразделение’ получается в результате расчета функции ‘Первое торговое подразделение’.

До вызова процедуры локальные переменные не существуют, они автоматически создаются именно в момент вызова. В случае рекурсивного вызова процедуры (когда процедура вызывает сама себя) каждый раз будет заново создаваться весь стек локальных переменных (в том числе и формальные параметры).

Ниже приводится пример рекурсивного вызова процедуры.

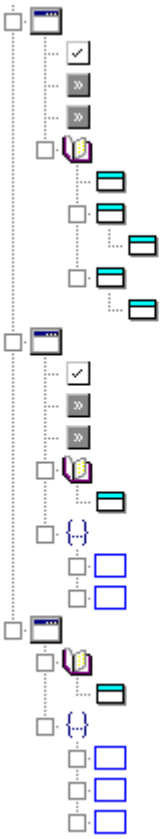


Процедура ‘Печать всех связанных документов’ просматривает связанные документы и выводит их UID-ы в log-файл. Документ считается связанным с текущим, если в поле ‘Родитель’ связанного документа записан UID текущего документа. Или такую связь можно назвать родитель-наследник. У наследника, в свою очередь, также могут быть связанные с ним документы. Процедура печатает наследников всех уровней от интересующего документа.

Разберем подробнее алгоритм процедуры.

UID интересующего документа передается в процедуру через формальный параметр. Перебираются все документы, связанные с интересующим, т.е. непосредственные наследники (или наследники первого уровня). Для каждого наследника первого уровня вызывается эта же процедура, но в качестве формального параметра ей передается UID наследника. Таким образом будут просмотрены все документы (их можно считать наследниками второго уровня по отношению к родителю), связанные с наследниками первого уровня. И так далее до самых последних наследников. Цикл завершится тогда, когда связанных документов больше не будет найдено.

Если процедуры связаны цепочкой наследования, то локальные переменные создаются одновременно для всей цепочки. Поэтому допустимо из процедуры-наследника обращаться к локальным переменным процедуры-родителя.



Первая процедура является базовой. Вторая процедура ссылается на базовую процедуру. В этой процедуре доступны как локальные переменные, описанные в самой процедуре, так и переменные из базовой процедуры. В третьей процедуре, поскольку она является наследником второй процедуры, можно использовать локальные переменные всех трех процедур. Переменная ‘Скидка’ описана в базовой процедуре. Описание переменной ‘Количество1’ находится во второй процедур, а переменная ‘Количество2’ указана как локальная для третьей процедуры.

Формальные параметры

В разделе ‘Формальные параметры’ располагаются описания параметров для вызова процедуры. По сути формальные параметры являются локальными переменными, и использовать их можно точно также, как и обычные локальные переменные. Отличие заключается в том, что начальное значение формальным параметрам можно задать двумя способами. В первом способе начальное значение указывается, как и для обычных локальных переменных, с помощью атрибута ‘начальное значение’. Во втором способе начальные значения формальных параметров задаются при вызове процедуры в разделе ‘Значения формальных параметров при вызове’.

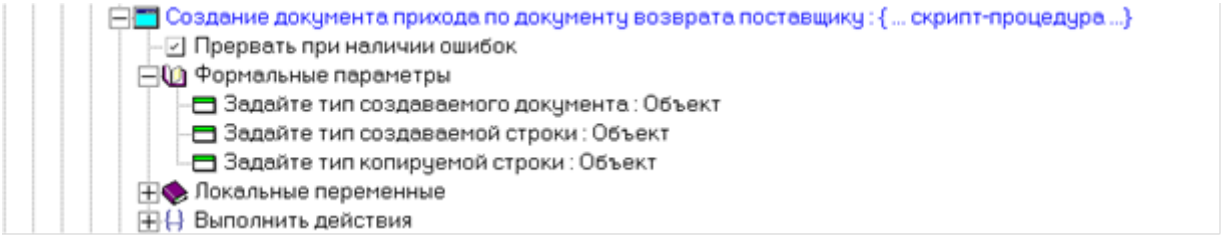
Значения переменных, указанные при вызове процедуры, имеют приоритет над начальными значениями в описании. Т.е. при старте процедуры в локальную переменную будет записано значение при вызове, а если таковое отсутствует, то будет записано начальное значение из описания.

Процедура с формальными параметрами в большей степени универсальная чем процедура

без параметров. Другие разработчики смогут применить такую процедуру, экономя свой труд. Кроме того, наличие одной универсальной процедуры вместо нескольких частных процедур уменьшает размер проекта. Исправление ошибок и модификацию алгоритма проще выполнить в одной процедуре, чем в нескольких.

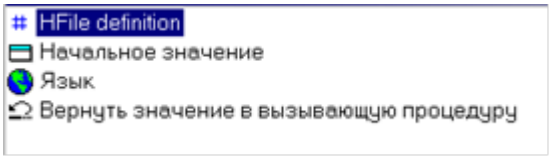
Рекомендуется выносить в качестве формальных параметров используемые в программе типы, классы, счета, кодификаторы. Для этого формальный параметр должен иметь тип ‘Объект’.

Формальные параметры также применяют для передачи UID-ов объектов. В этом случае при вызове процедуры для получения нужного объекта указывают различные выражения. Тип формального параметра может быть либо ‘Объект’, либо соответствовать типу объекта (‘Каталог’, ‘Продукт’, ‘Документ’ и т.д.).



Процедура имеет три формальных параметра типа ‘Объект’.

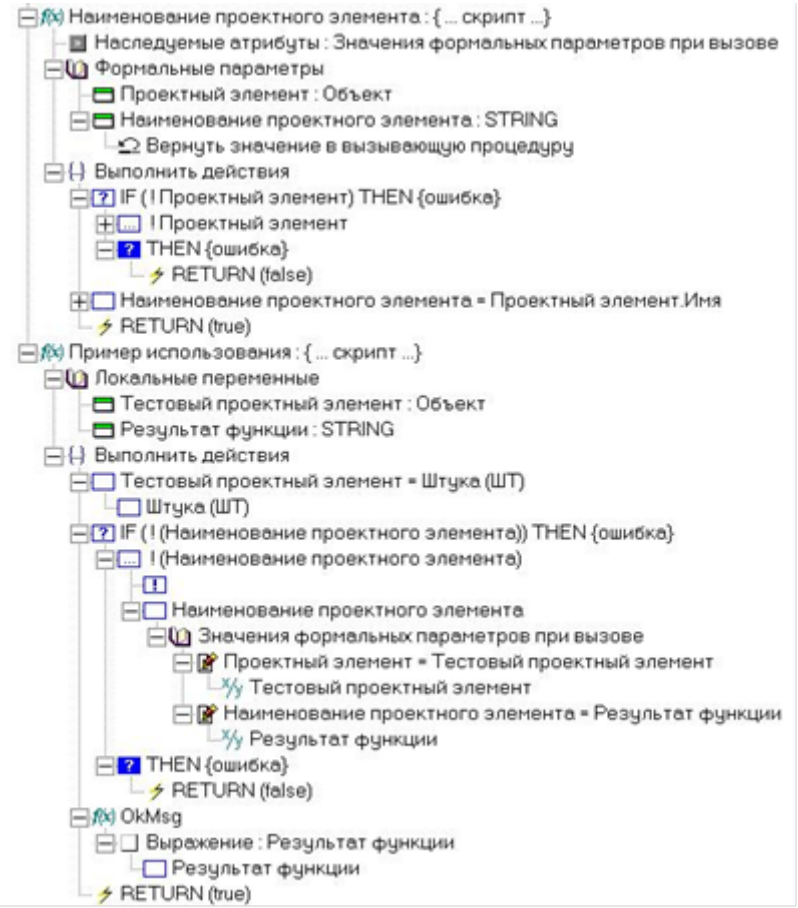
Для каждого формального параметра можно указать следующие атрибуты:



- **Начальное значение –** это значение будет присвоено формальному параметру, если не будет указано иное значение в разделе ‘*Значения формальных параметров при вызове*’.
- **Вернуть значение в вызывающую процедуру** – наличие этого признака означает, что значение параметра должно быть возвращено в вызывающую процедуру. Значение параметра может быть возвращено только в том случае, если в качестве параметра при вызове указана локальная или глобальная переменная (а не выражение или функция).

Таким образом, в языке скриптов реализована модель IN-OUT параметров (не путать с параметрами, передаваемыми по ссылке). Возврат значения в вызывающую процедуру происходит в момент выхода из вызываемой процедуры, но не в момент присвоения значения формальному параметру внутри вызываемой процедуры.

Для лучшего понимания рассмотрим условный пример.

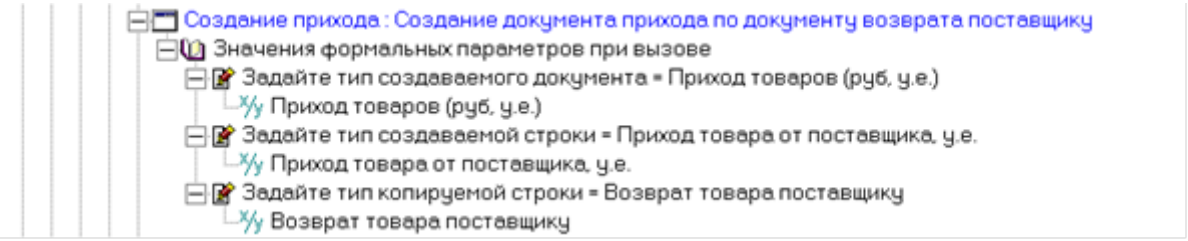


Функция ‘Наименование проектного элемента’ имеет два формальных параметра: ‘Проектный элемент’ и ‘Наименование проектного элемента’. Функция возвращает истинное значение, если удалось получить наименование элемента, или ложное значение, если переданный UID принадлежит не проектному элементу. Первый параметр используется для передачи значения в функцию, во втором параметре возвращается наименование. При вызове этой функции из основной процедуры заполняются оба параметра. Причем при заполнении второго параметра указана локальная переменная ‘Результат функции’. Значение в эту переменную записывается после выполнения функции и используется в дальнейшем тексте основной процедуры.

- **Язык** – используется для поддержки национальных языков. Определяет преобразование данных, которое будет выполняться при выводе строки на экран и при вводе ее с клавиатуры.

Значения формальных параметров при вызове

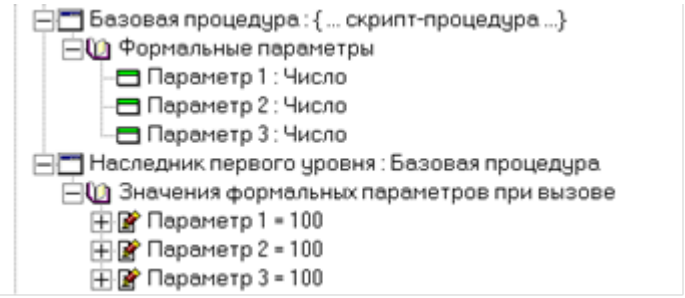
В этом разделе указываются значения формальных параметров при вызове процедуры.



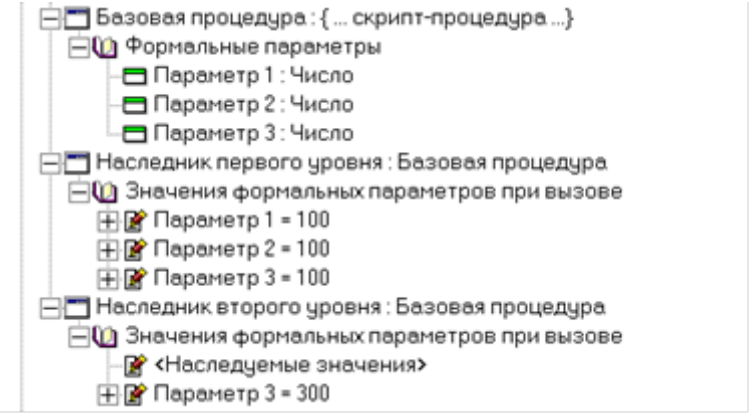
Вызов процедуры с заполнением формальных параметров. Выражения ссылаются на проектные элементы. Описание процедуры находится на предыдущем рисунке.

> При наличии цепочки наследования процедур указывать значения формальных параметров можно на любом уровне наследования. Для наследования значений применяется соответствующий атрибут.

Примеры:



Две процедуры связаны цепочкой наследования. Процедура-родитель имеет три формальных параметра. При описании процедуры-наследника заполнены все три параметра.

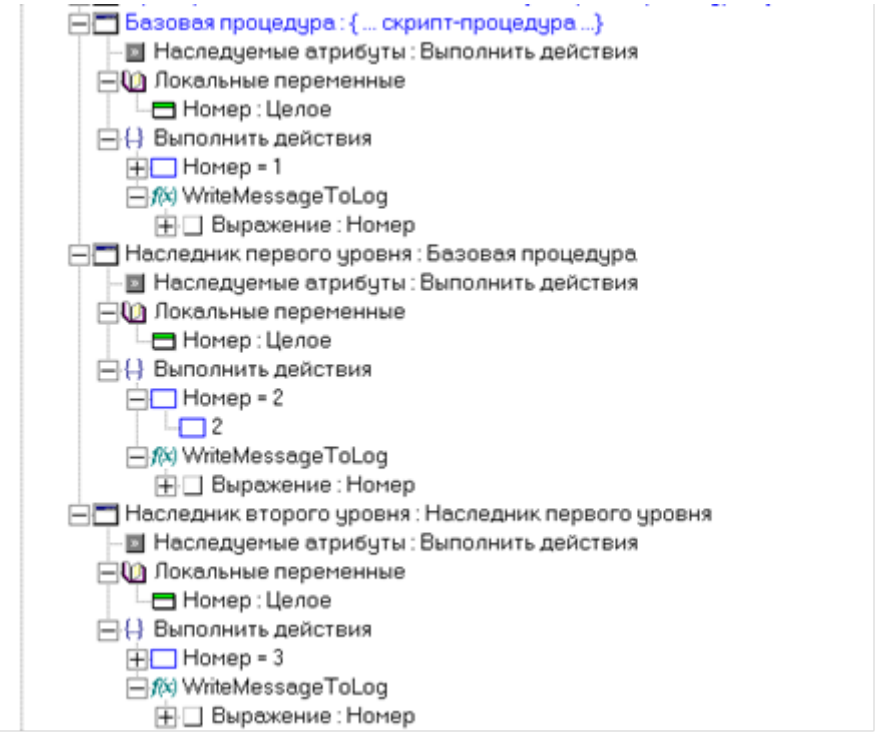


При описании процедуры-наследника второго уровня изменено значение третьего параметра.

Выполнить действия

Раздел ‘Выполнить действия’ содержит основной блок (тело) процедуры. Последовательность операторов, содержащаяся в этом блоке, автоматически выполняется

при запуске процедуры. Если процедуры связаны цепочкой наследования, то выполнение этого блока идет в направлении от ребенка к родителю, т.е. снизу вверх.

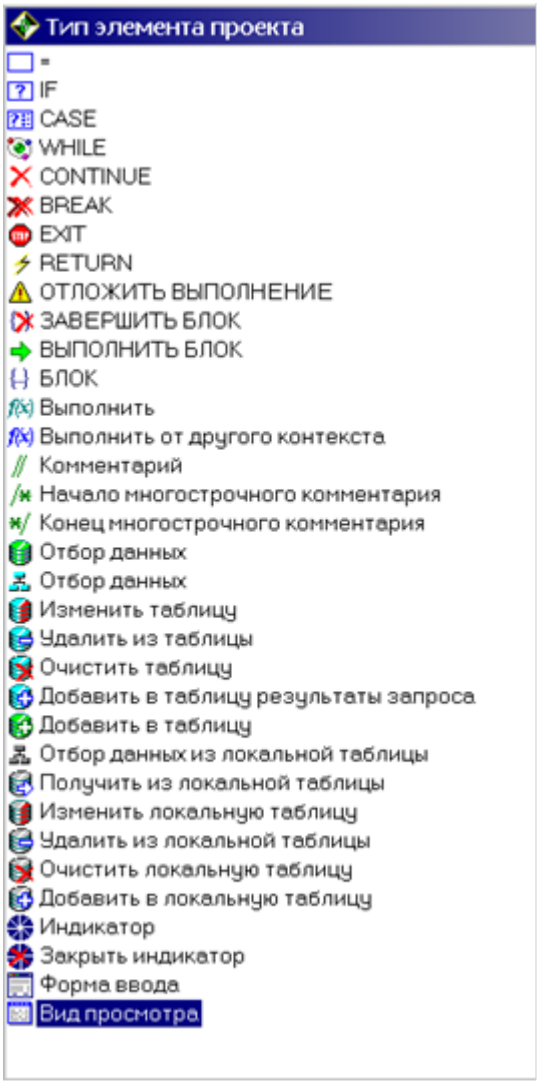


Три процедуры связаны цепочкой наследования. В каждой процедуре заполняется переменная ‘Номер’. Затем значение этой переменной заносится в log-файл. Последовательность выполнения будет следующей: сначала выполнится ‘Наследник второго уровня’, затем - ‘Наследник первого уровня’, и наконец – ‘Базовая процедура’.

В log-файле последовательно будут записаны три числа: 3,2,1.

В цепочках наследования рекомендуется использовать раздел ‘Выполнить действия’ либо только у родителя, либо только у наследника. В этом случае легче понять алгоритм работы процедуры.

Для описания тела процедуры можно использовать следующие операторы языка скриптов:



Операторы *‘Присвоить’, ‘IF’, ’CASE’, ‘WHILE’, ‘CONTINUE’, ‘BREAK’, ‘EXIT’, ‘RETURN’, ‘ОТЛОЖИТЬ ВЫПОЛНЕНИЕ’, ‘ЗАВЕРШИТЬ БЛОК’, ‘ВЫПОЛНИТЬ БЛОК’, ‘БЛОК’, ‘Выполнить’ (Вызов процедуры или функции), ‘Выполнить от другого контекста’, ‘Комментарий’, Начало и конец многострочного комментария, ‘Индикатор’, ‘Закрыть индикатор’, ‘Форма ввода’, ‘Вид просмотра’* рассматривались ранее.

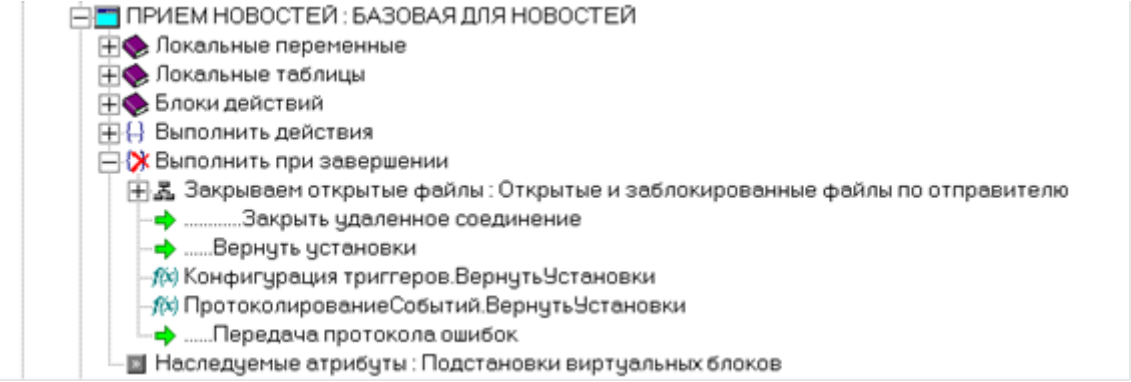
В дальнейшем тексте приведено подробное описание операторов как для работы с таблицами *(‘Отбор данных’ в двух вариантах, ‘Изменить таблицу’, ‘Удалить из таблицы’, ‘Очистить таблицу’, Добавить в таблицу результаты запроса’, ‘Добавить в таблицу’),* так и для обслуживания локальных таблиц *(‘Отбор данных из локальной таблицы’, ‘Получить из локальной таблицы’, ‘Изменить локальную таблицу’, ‘Удалить из локальной таблицы’, ‘Очистить локальную таблицу’, ‘Добавить в локальную таблицу’).*

Выполнить при завершении

Это блок операторов автоматически выполняется при завершении процедуры после основного блока. Блок ‘Выполнить при завершении’ запускается независимо от того, как завершился основной блок – нормально или был прерван.

Обычно блок ‘Выполнить при завершении’ применяют для закрытия открытых в основном

блоке ресурсов (файлов, выборок по файлам и т.д.). Процедуры закрытия ресурсов гарантировано выполнятся, даже если основной блок процедуры был прерван по ошибке или по воле пользователя. Аналогично основному блоку, если процедуры связаны цепочкой наследования, то выполнение этого блока идет в направлении от ребенка к родителю, т.е. снизу вверх.

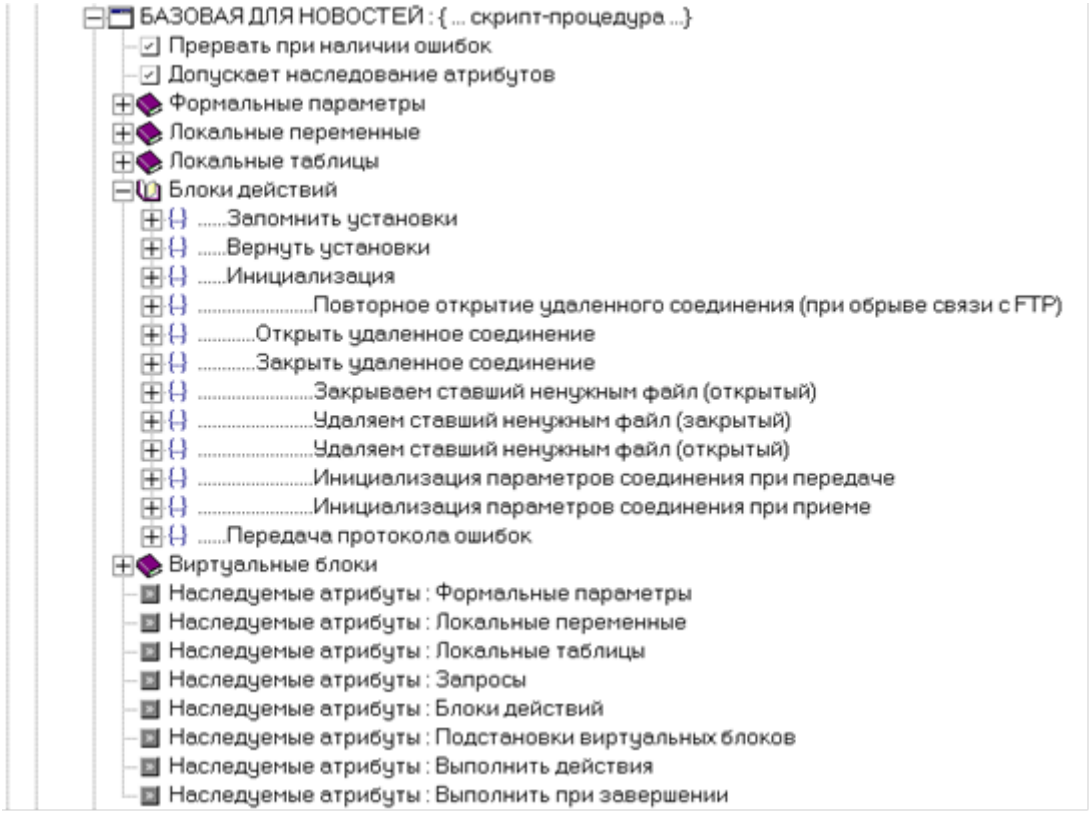


> В цепочках наследования рекомендуется использовать раздел ‘Выполнить при завершении’ либо только у родителя, либо только у наследника. В противном случае понимание текста процедуры может быть затруднено.

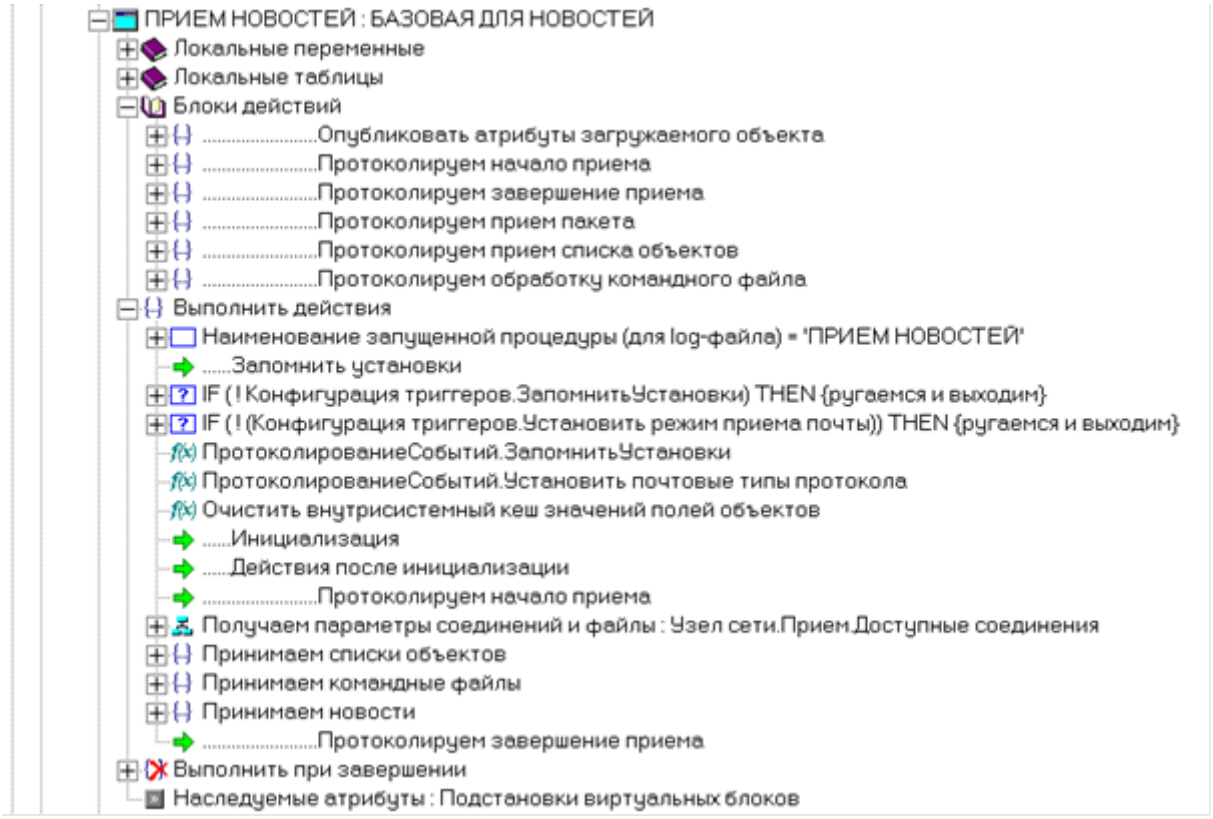
Блоки действий

Здесь описываются блоки команд (операторов), вынесенные из основного блока (тела) процедуры. Блоки, описанные в этом разделе, в отличии от блоков тела процедуры, не вызываются автоматически, а требуют явного вызова. Вызывать блоки можно не только из тела процедуры, но и из процедур-наследников.

Процедуры, грамотно разбитые на блоки, выглядят в проекте понятно и красиво. Посмотрим на структуру большой и сложной процедуры.



Базовая процедура не имеет тела, но описаны несколько блоков действий.



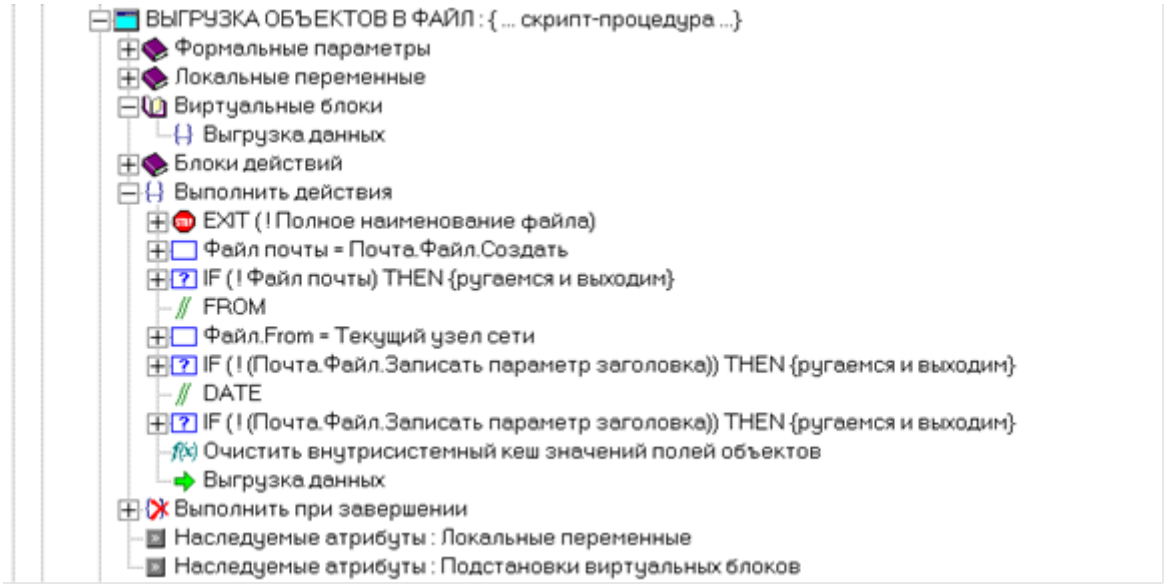
В процедуре-наследнике описаны еще несколько блоков действий. В основном блоке вызываются как блоки из базовой процедуры (‘...Инициализация’), так и блоки из процедуры-наследника (‘...Протоколируем начало приема’).

Виртуальные блоки

В данном разделе перечисляются виртуальные блоки процедуры. Виртуальные блоки используются для переноса части логики в процедуру-наследник. В теле процедуры имеются ссылки на виртуальные блоки, но свое содержимое виртуальные блоки получают только в процедуре-наследнике в разделе ‘*Подстановки виртуальных блоков*’.

Обычно в виртуальных блоках содержатся действия, выполняемые при старте и завершении процедуры, расчет различных для каждого из наследников выражений, произвольная фильтрация данных внутри базовой процедуры и т.д.

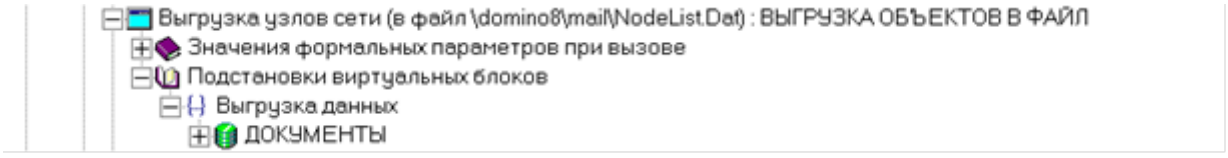
Пример процедуры, использующей виртуальные блоки:

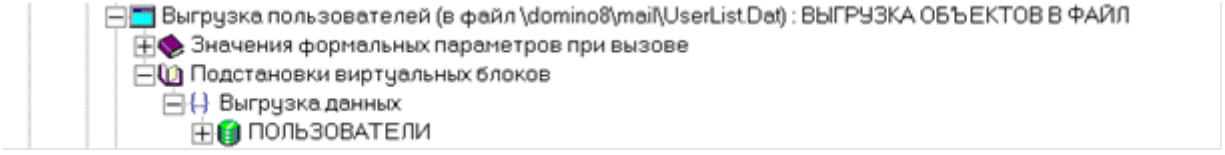


Процедура содержит один виртуальный блок ‘Выгрузка данных’. В теле процедуры имеется вызов этого блока.

Подстановки виртуальных блоков

В этом разделе процедуры-наследника виртуальные блоки получают свое содержимое в виде последовательности команд (операторов). Для этого описывается блок действий со ссылкой на виртуальный блок.





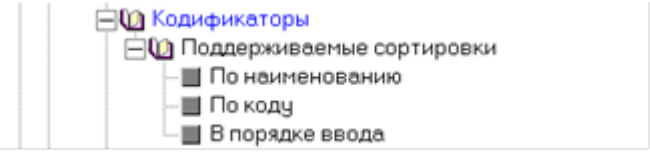
Обе процедуры являются наследниками базовой процедуры, описание которой находится на предыдущем рисунке. Первая процедура выгружает данные из таблицы

Документы, вторая процедура – из таблицы Пользователи. В соответствии с предназначением заполняется содержимое виртуального блока.

Если несколько процедур, связанных цепочкой наследования, переопределяют один и тот же виртуальный блок, то выполняться будет содержимое виртуального блока, описанного на самом нижнем уровне.

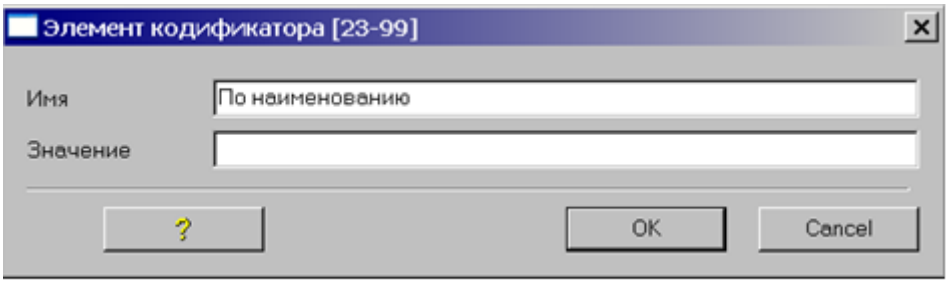
Кодификаторы

Кодификатор содержит список именованных проектных элементов. Обычно кодификатор применяется для выбора пользователем одной альтернативы из списка возможных вариантов.



Кодификатор содержит три элемента.

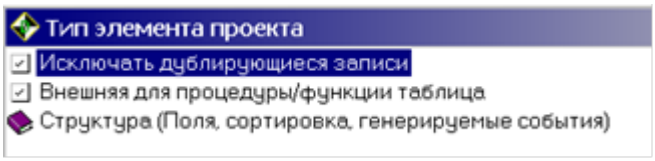
Для каждого элемента кодификатора можно указать имя и значение.



Локальные таблицы

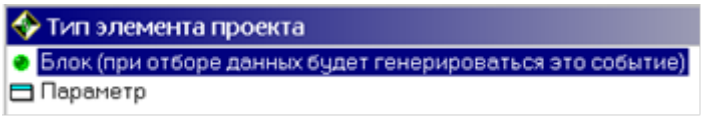
Локальные таблицы применяются для хранения промежуточных результатов и передачи данных из одной процедуры в другую. Локальная таблица описывается и заполняется внутри процедуры. Для таблицы можно указать произвольные сортировку и группировку записей.

Раздел ‘Локальные таблицы’ содержит перечень локальных таблиц процедуры. Для каждой таблицы можно указать имя, список полей, условия сортировки и группировки записей и два специальных признака.



- **Структура** (**Поля, сортировка, генерируемые события)** – содержит список параметров (полей) локальной таблицы. При описании полей таблицы используются

локальные, глобальные и контекстные переменные. Тип поля таблицы определяется типом соответствующей переменной. Значения указанных переменных сохраняются в строке таблицы при добавлении и изменении записи. При выборке записей из таблицы переменные заполняются считанными значениями. Таким образом, можно считать, что запись (строка) таблицы – это множество всех переменных, перечисленных в описании таблицы.



Для каждого поля таблицы можно указать вид сортировки таблицы по этому полю: по убыванию значения поля или по возрастанию значения. Наличие признака сортировки у поля означает, что данное поле входит **в ключ сортировки**. Если признак сортировки указан у нескольких полей, то ключ сортировки является составным. Порядок полей в ключе соответствует порядку описания полей таблицы.

Если для таблицы не указан ключ сортировки, то записи в таблицы сохраняются в хронологическом порядке их добавления.

Если ключ сортировки не является уникальным, то записи таблицы сортируются в соответствии с ключом. При добавлении записи с повторяющимся значением ключевых полей эта запись располагается после всех имеющихся записей с таким же значением ключа.

При уникальном ключе сортировки все записи располагаются строго в указанном порядке. Повторяющиеся записи исключены.

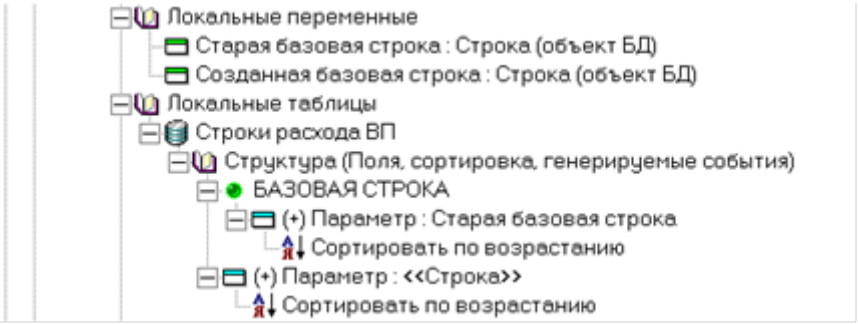


Таблица содержит два поля типа ‘Объект’. Первое поле ссылается на локальную переменную ‘Старая базовая строка’. Второе поле ссылается на контекстную переменную ‘<<Строка>>’. Оба поля являются ключевыми. Записи таблицы будут отсортированы по возрастанию значения поля ‘Старая базовая строка’. Те записи, у которых значение этого поля совпадает, будут отсортированы по возрастанию значения поля ‘<<Строка>>’.

Для группировки записей таблицы по ключевым полям применяется элемент проекта ‘Блок (при отборе данных будет генерироваться это событие)’. В блоке объединяются записи с одинаковыми значениями тех ключевых полей, описание которых находится внутри блока.

Последовательность расположения блоков в описании таблицы определяет уровень

(вложенность) группировки записей.

В примере на рисунке выше описан один блок ‘Базовая строка’. Внутри этого блока находится описание поля таблицы ‘Старая базовая строка’.

При отборе данных из локальной таблицы отслеживается изменение ключевых полей. Как только поменяется условие группировки блока (значения ключевых полей блока), программа для первой записи с новыми значениями ключевых полей генерирует событие ‘*Выполнить перед обработкой блока’*, а после обработки последней записи с таким же значением ключа – событие ‘*Выполнить после обработки блока’*.



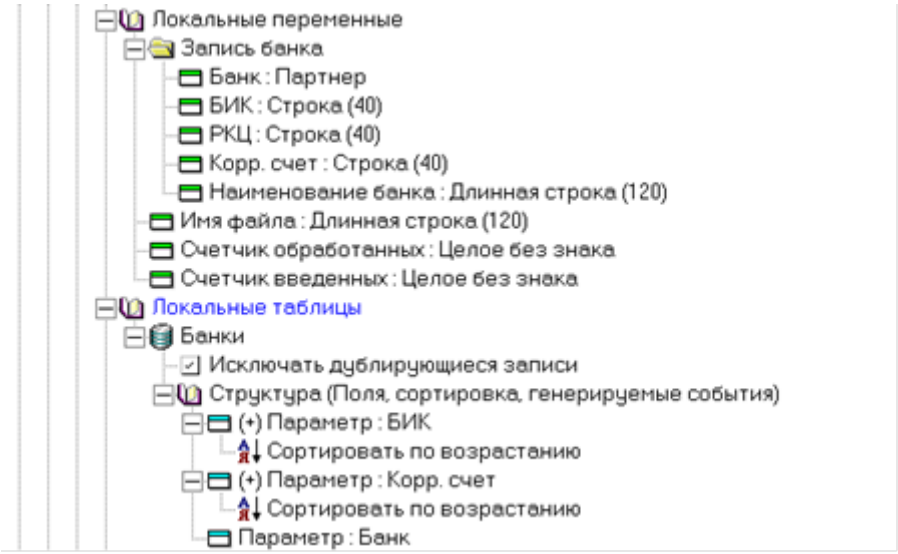
Для рассматриваемого примера вызов этих событий будет выглядеть таким образом.

Самый нижний уровень группировки (поля вне блоков или таблица без блоков) попадает в одно событие ('*Выполнить после получения записи*'), которое генерируется для каждой новой записи при отборе данных (после события ‘*Выполнить перед обработкой блока’*, но перед событием ‘*Выполнить после обработки блока’*).



Описание таблицы может не содержать блоков. Если таблица содержит блоки, то они должны содержать ключевые поля, по котором производится группировка данных в этих блоках. Иными словами, если в описании блока заданы поля, то хотя бы одно их них должно быть ключевым. Описание блока без ключевых полей не является синтаксической ошибкой, но такой блок не влияет на генерацию событий.

- **Исключать дублирующие записи** – данный признак применяется для получения таблицы с уникальными, неповторяющимися записями. Уникальность проверяется по ключу сортировки . Ключ сортировки – это набор значений полей таблицы, для которых указан признак сортировки. При установленном признаке попытка добавить в таблицу запись, содержащую повторный ключ сортировки, отвергается.

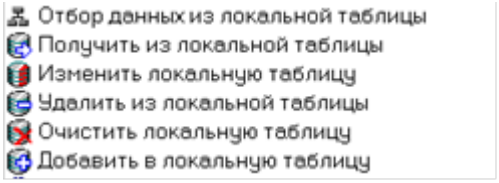


Локальная таблица содержит список банков. Записи в таблице не повторяются. Проверка на уникальность проводится по двум параметрам: ‘БИК’ и ‘Корр.счет’.

- **Внешняя для процедуры/функции таблица** – данный признак используется для передачи таблицы между процедурами (в том числе при рекурсивном вызове процедуры). Наличие признака означает, что будучи раз созданной, таблица при завершении процедуры не удаляется. В случае рекурсивного вызова процедуры внешняя локальная таблица не пересоздается. Данные в такой таблице сохраняются в течении всей сессии (до выхода из программы или перерегистрации пользователя).

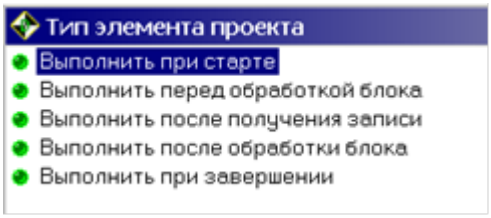
Наличие признака ‘внешняя’ для локальной таблицы не означает, что такая таблица будет видна во всех процедурах и функциях. Область видимости локальной таблицы рассчитывается по обычным для проекта Домино правилам. Локальная таблица доступна внутри той процедуры, где она была определена, и внутри процедур-наследников.

Далее рассмотрим операторы для работы с локальными таблицами:



Оператор Отбор данных из локальной таблицы

Оператор ‘Отбор данных из локальной таблицы’ применяется для последовательного считывания записей. В процессе считывания записей вызываются блоки действий, соответствующие следующим событиям:



- **Выполнить при старте -** Последовательность операторов, которые будут выполнены сразу после получения первой записи из таблицы. Если данных нет, то блок ‘*Выполнить при старте*’ не вызывается.
- **Выполнить перед обработкой блока -** Операторы, которые будут выполнены после получения следующей записи, если изменились значения ключевых полей, описанных в блоке. При получении из таблицы первой записи блок ‘*Выполнить перед обработкой блока*’ вызывается сразу после вызова блока ‘*Выполнить при старте*’.
- **Выполнить после получения записи - О**ператоры, которые будут выполнены после получения каждой следующей записи из таблицы. Если больше записей нет, то блок ‘*Выполнить после получения записи*’ не вызывается.

При получении **первой** записи блок ‘*Выполнить после получения записи*’ вызывается **после** вызова блока ‘*Выполнить при старте*’ и всех блоков ‘*Выполнить перед обработкой блока*’. При получении **последней** записи блок ‘*Выполнить после получения записи*’ вызывается **перед** вызовом всех блоков ‘*Выполнить после обработки блока*’ и блока ‘*Выполнить при завершении*’.

- **Выполнить после обработки блока - О**ператоры, которые будут выполнены после получения последней записи с текущими значениями описанных в блоке ключевых полей.

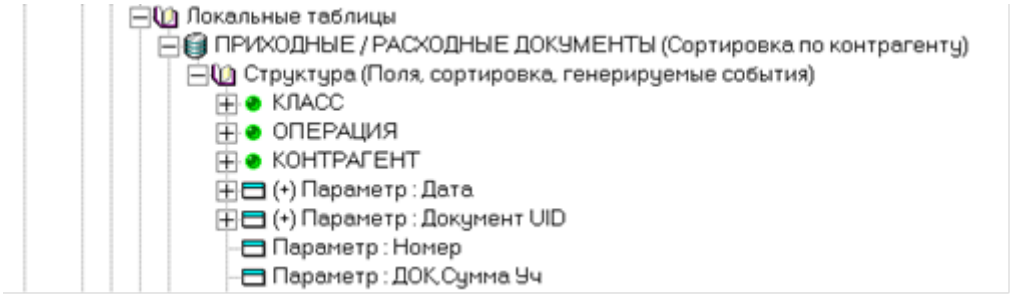
При получении из таблицы последней записи блок ‘*Выполнить после обработки блока*’ вызывается перед вызовом блока ‘*Выполнить при завершении*’.

- **Выполнить при завершении** - Операторы, которые будут выполнены после получения последней записи из таблицы. Если записей не было, то блок ‘*Выполнить при завершении*’ не вызывается.

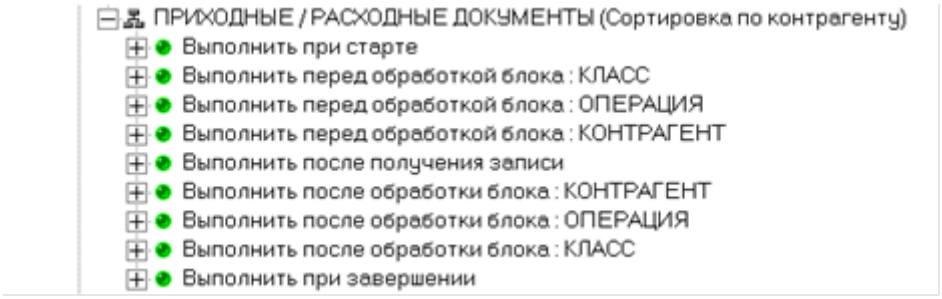
Одновременное указание всех блоков не обязательно, и определяется логикой программы.

После завершения считывания записей из таблицы управление возвращается в процедуру.

Примеры.



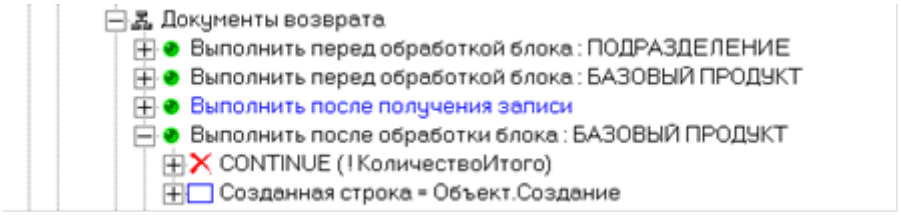
Описание локальной таблицы.



Максимально возможное количество вызовов блоков-событий для данной таблицы.

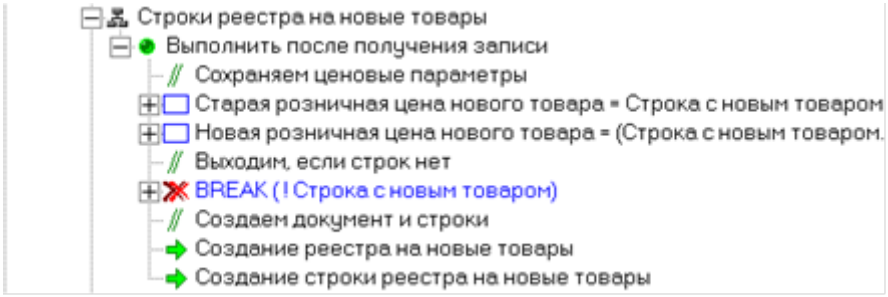
При отборе данных можно использовать операторы ‘CONTINUE’, ‘BREAK’ и ‘*Завершить блок действий*’.

Оператор ‘CONTINUE’ вызывает переход к получению следующей записи. Если оператор ‘CONTINUE’ вызвать из блока ‘*Выполнить при старте*’, то выборка данных прервется. Блок '*Выполнить при завершении*’ в этом случае не вызывается.



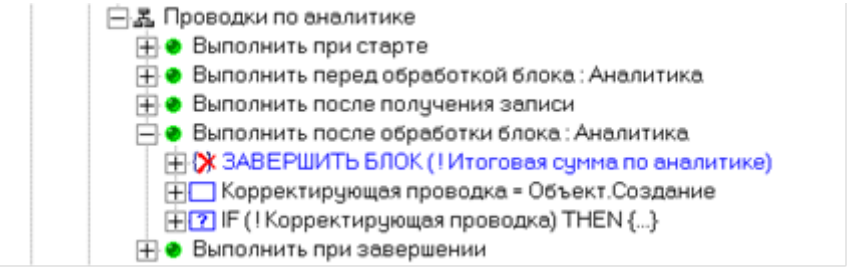
Если итоговое количество по продукту равно 0, то обработка текущего блока прерывается, и считывается следующая запись.

Оператор ‘BREAK’ прерывает отбор данных. Блок ‘*Выполнить при завер*шении’ не вызывается.



Если полученная строка не содержит новый товар, то считывание записей из таблицы прерывается. Блок ‘Выполнить при завершении’ отсутствует.

Оператор ‘*Завершить блок действий*’ завершает выполнение текущего блока-события. Отбор данных не прерывается, и последовательность вызова очередных блоков-событий не меняется.



Если итоговая сумма по аналитике равна 0, то обработка текущего блока прерывается

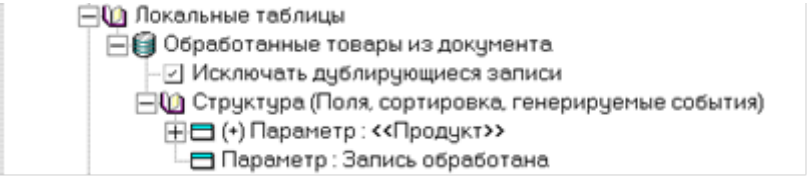
При отборе данных из локальной таблицы изменение значений описанных в блоке ключевых полей определяет два события: вход и выход из блока. Последний уровень (поля вне блоков) попадает в событие ‘*Выполнить после получения записи*’, которое генерируется для каждой новой записи при отборе данных.

Оператор Получить из локальной таблицы

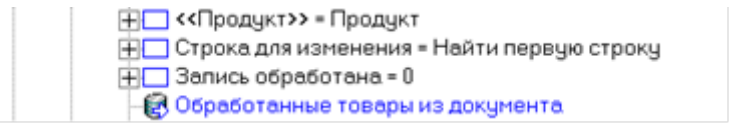
Оператор ‘*Получить из локальной таблицы*’ применяется для считывания записи из локальной таблицы. Для поиска записи используются текущие значения ключевых полей.

Если запись найдена, то заполняются все поля записи (точнее, заполняются значения всех не ключевых полей, а значения ключевых переменных не изменятся). Если запись не найдена, то значения всех не ключевых полей будут сброшены в NULL (пусто), а значения ключевых полей останутся без изменения.

Если в таблице допустимы записи с повторяющимися значениями ключевых полей, то возвращается любая из записей с текущим значением ключа.



Описание таблицы.



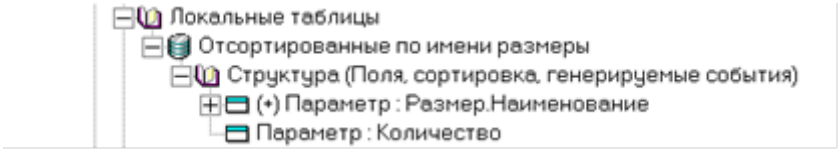
Перед вызовом оператора для получения записи из таблицы заполняются переменные ‘<<Продукт>>’ и ‘Запись обработана’.

Оператор ‘*Получить из локальной таблицы*’ допустимо вызывать из выборки данных по той же таблице, при этом указатель текущей записи выборки не сбивается.

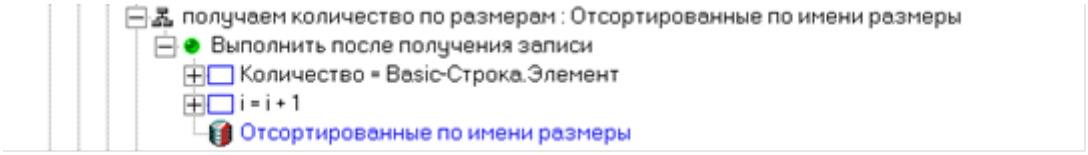
Оператор Изменить локальную таблицу

Данный оператор изменяет текущую запись локальной таблицы. Оператор допустимо вызывать только если установлен указатель текущей записи. Указатель текущей записи устанавливается либо при отборе записей из этой таблицы, либо после вызова оператора ‘*Получить из локальной таблицы*’. Если указатель текущей записи не установлен, то никакие действия не выполняются.

Если значения ключевых полей соответствуют значению ключа текущей записи, то изменяется текущая запись. В противном случае текущая запись удаляется и добавляется новая запись. Значения полей записи берутся из переменных, которые были использованы при описании таблицы. При добавлении к таблице записи автоматически сортируются по ключу. Записи с одинаковым значением ключа записываются в хронологическом порядке. Если для таблицы указан признак ‘*Исключать дублирующие записи*‘, то попытка добавить запись с повторяющимся значением ключа отвергается. При этом выполнение процедуры не прерывается, и сообщение об ошибке не выдается.



Описание таблицы.



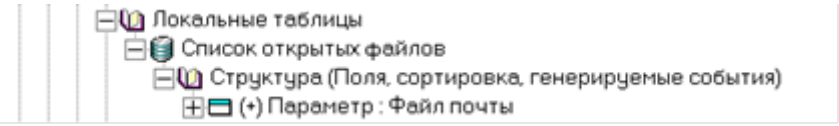
Перебираются записи из локальной таблицы. Перед изменением записи уменьшается значение поля ‘Количество’. В блоке ‘Выполнить после получения записи’ указатель текущей записи устанавливается автоматически.

С помощью оператора ‘Изменить локальную таблицу’ допустимо изменять записи в таблице, по которой идет выборка данных, при этом указатель текущей записи выборки не сбивается.

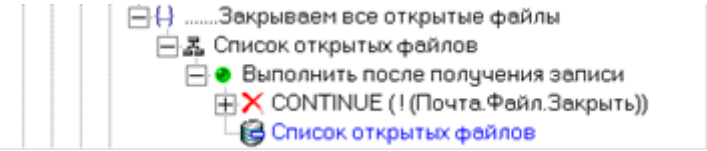
Оператор Удалить из локальной таблицы

Данный оператор удаляет запись из локальной таблицы. Для поиска записи используются текущие значения всех ключевых полей. Если запись не найдена, то никакие действия не

выполняются.



Описание таблицы.

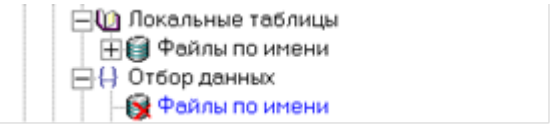


В блоке действий просматривается список открытых файлов. Для каждого файла запускается функция, закрывающая файл. Если файл удалось закрыть без ошибок, то запись о файле удаляется из локальной таблицы. В момент удаления записи в поле таблицы находится значение, заполненное в блоке 'Выполнить после получения записи'.

Допустимо удалять записи из таблицы, по которой идет выборка данных, при этом указатель текущей записи выборки не сбивается.

Оператор Очистить локальную таблицу

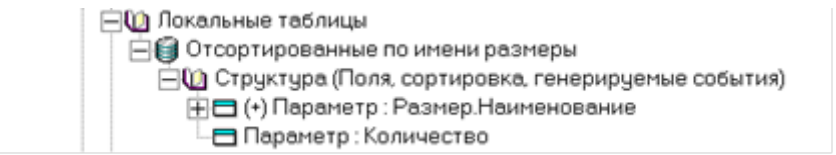
Оператор '*Очистить локальную таблицу'* удаляет все записи из локальной таблицы.



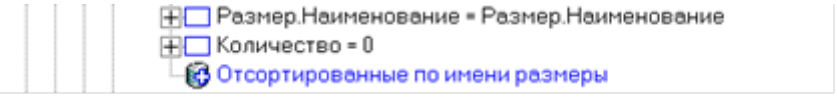
Блок начинается с очистки локальной таблицы.

Оператор Добавить в локальную таблицу

Данный оператор добавляет запись в локальную таблицу. Значения полей записи берутся из переменных, которые были использованы при описании таблицы. При добавлении к таблице записи автоматически сортируются по ключу. Записи с одинаковым значением ключа записываются в хронологическом порядке. Если для таблицы указан признак ' *Исключать дублирующие записи*', то попытка добавить запись с повторяющимся значением ключа отвергается. При этом выполнение процедуры не прерывается, и сообщение об ошибке не выдается.



Описание таблицы.



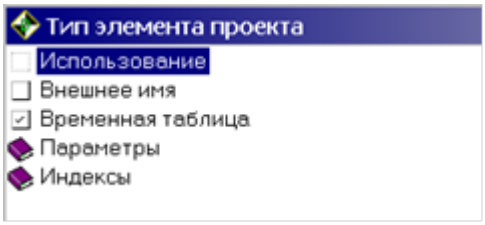
Перед вызовом оператора для добавления записи в таблицу заполняются переменные ‘Размер.Наименование’ и ‘Количество’.

Допустимо добавлять записи к таблице, из которой идет выборка данных, при этом указатель текущей записи выборки не сбивается. Однако, необходимо учитывать возможность ‘зациклить’ отбор данных. Это может произойти, если добавление записи происходит ‘вперед’ (т.е. значения ключевых полей новой записи больше значений ключевых полей текущей записи выборки). Запись, добавленная вперед, рано или поздно будет отобрана. В процессе ее обработки будет порождена следующая новая запись ‘вперед’, которая также будет отобрана, и, в свою очередь, породит запись. В таблице будут появляться новые записи до тех пор, пока диск не переполнится громадным файлом локальной таблицы.

Таблицы

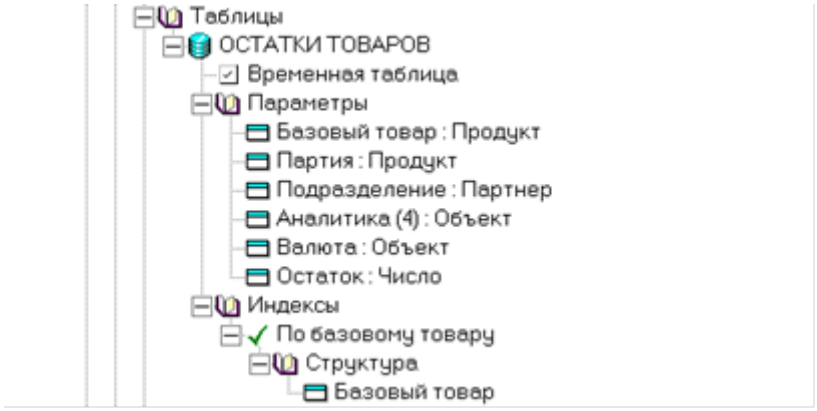
Раздел ‘Таблицы’ содержит перечень используемых таблиц на сервере БД. В описании таблицы можно указать ее наименование на сервере, набор полей и набор индексов. При старте программы проверяется существование таблицы. Если таблица отсутствует, то программа создает таблицу в соответствии с описанием таблицы в проекте. Если таблица уже имеется, то программа проверяет описание всех полей таблицы. В случае расхождения описание поля в таблице сервера БД изменяется в соответствии с описанием в проекте.

При описании таблицы доступны следующие атрибуты:



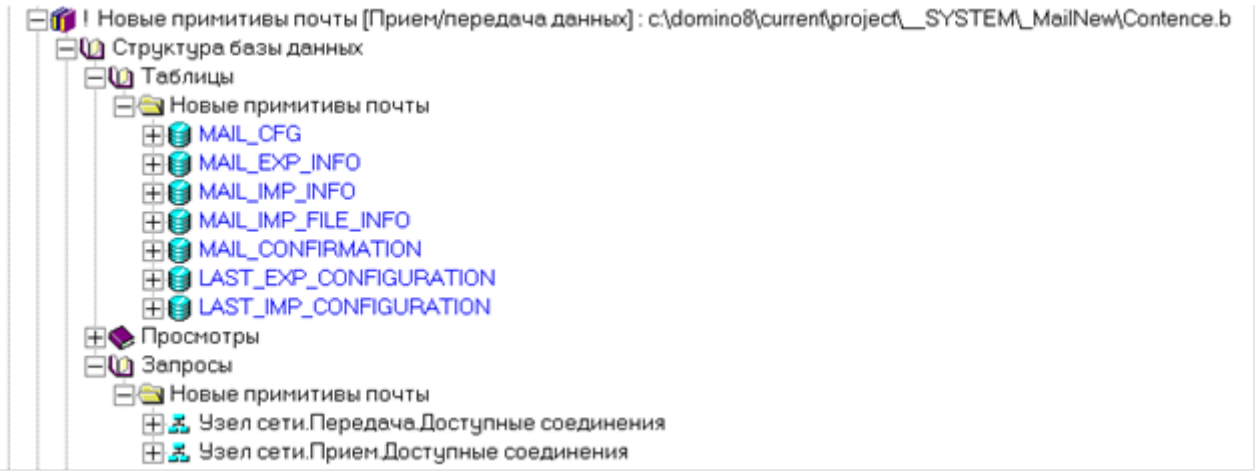
- **Использование** – условие доступа проектировщиков к таблице.
- **Внешнее имя** – имя таблицы в базе данных. Если внешнее имя не указано, то имя таблицы рассчитывается автоматически (T<индекс проектного элемента>).
- **Временная таблица** – таблица, для которой указан данный признак, будет автоматически удалена сервером при выходе из Домино (или при перерегистрации пользователя).
- **Параметры** – список полей (столбцов) таблицы. Для каждого поля указываются имя и тип. Можно указать внешнее имя (имя столбца в базе данных), ширину и точность для числовых данных. Если внешнее имя не указано, то имя столбца рассчитывается автоматически (F<индекс проектного элемента>).
- **Индексы** – список индексов таблицы. Индексы применяются для ускорения запросов

по данной таблице. Для каждого индекса указываются список полей.



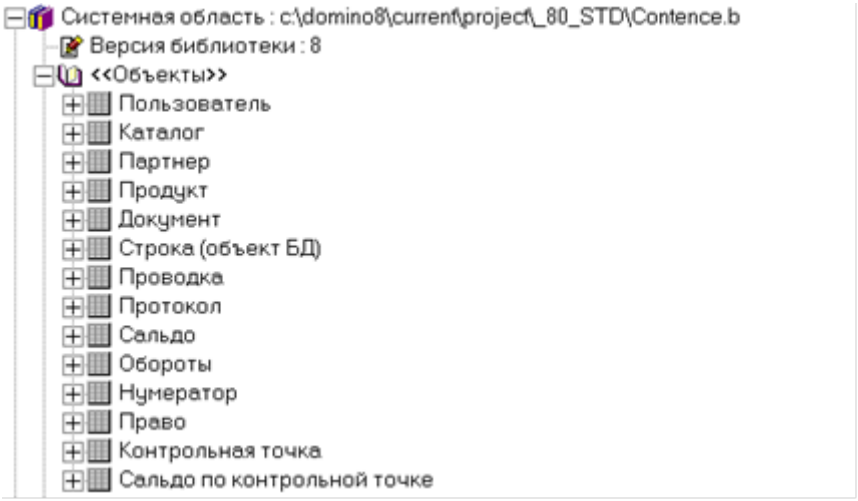
Временная таблица ‘Остатки Товаров’ состоит из шести полей. Для таблицы указан индекс по базовому товару.

Таблицы могут быть описаны не только в разделе ‘Таблицы’ для процедур. В дереве проекта имеется специальный раздел для описания таблиц и запросов.

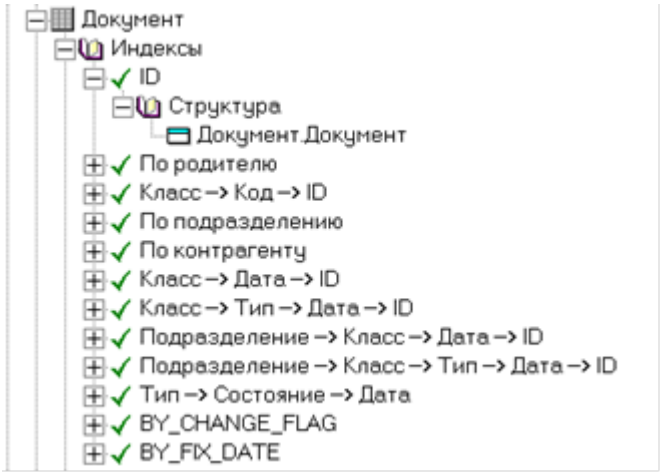


Библиотека ‘Новые примитивы почты’ содержит раздел ‘Структура базы данных’, в котором описаны таблицы для работы почтовой службы.

Основные таблицы БД описаны в библиотеке ‘Системная область’ в разделе ‘<<Объекты>>’. Их описание отличается от описания остальных таблиц.



Индексы основных таблиц БД описаны в библиотеке ‘Системная область’ в разделах, соответствующих объектам БД.



Раздел ‘Документ’ в папке ‘Индексы’ содержит список всех индексов по таблице ‘Документ’.

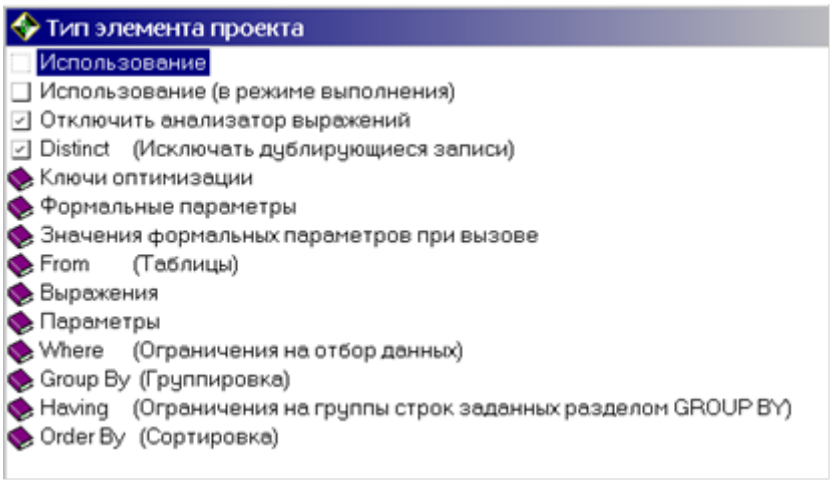
Запросы

В этом разделе описываются SQL запросы для выборки данных из таблиц БД. Описание запроса в ДОМИНО выполнено для стандарта SQL92. Для запросов можно применить механизмы наследования, запуск с параметрами, условия включения отдельных частей. Все это позволяет выделить общую базовую часть для сложных и часто используемых запросов. При создании запроса можно ссылаться на базовые запросы и указывать нужные параметры, сортировки, условия отбора и т.д.

Процесс выполнения запроса можно разделить на два этапа. Первый этап называется формированием (компиляцией) запроса. На этом этапе на основе описания запроса в проекте создается предложение на языке SQL. Второй этап заключается в выполнении запроса на сервере БД. Сформированное предложение передается серверу.

> Запросы могут быть вложенными, т.е. их можно указывать не только в разделе ‘*Запросы*’, но и внутри конструкций FROM, WHERE, IN, EXISTS, HAVING, в выражениях. При описании запроса можно сослаться на уже существующий запрос. В этом случае по правилам наследования становятся доступны атрибуты запроса-родителя.

Для описания запроса применяются следующие атрибуты:



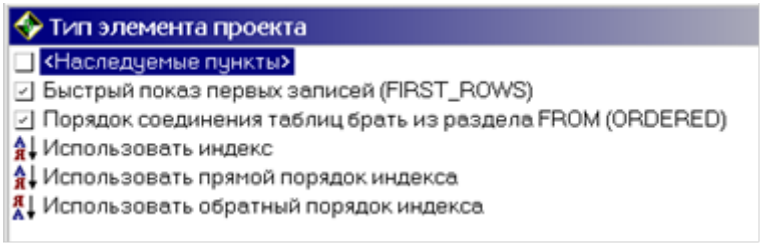
- **Использование** – условие доступа проектировщиков к запросу.
- **Использование (в режиме выполнения)** – условие доступа пользователей к запросу. Если условие имеет ложное значение, то запрос не будет сформирован.
- **Отключить анализатор выражений** – наличие данного признака отключает стандартный алгоритм формирования запроса. В стандартном алгоритме программа проверяет использование каждого параметра (из раздела ‘*Параметры*’). Если какой-либо параметр не используется ни в одной конструкции запроса, то такой параметр не будет включен в результат. В большинстве случаев стандартный алгоритм позволяет ускорить выполнение запроса. Наличие данного признака отключает анализ параметров запроса на предмет их реального использования, в результат будут включены все перечисленные в запросе параметры.

В практике применения Домино всего несколько раз встретились настолько сложные и специфичные запросы, что применяемый анализатор выражений не справился со своей задачей. В подобных ситуациях следует использовать данный атрибут.

- **Distinct (Исключать дублирующие записи)** – наличие этого признака исключает дублирующиеся значения из результата выполнения запроса.

Ключи оптимизации

Данный раздел содержит инструкции, позволяющие улучшить выполнение запроса.

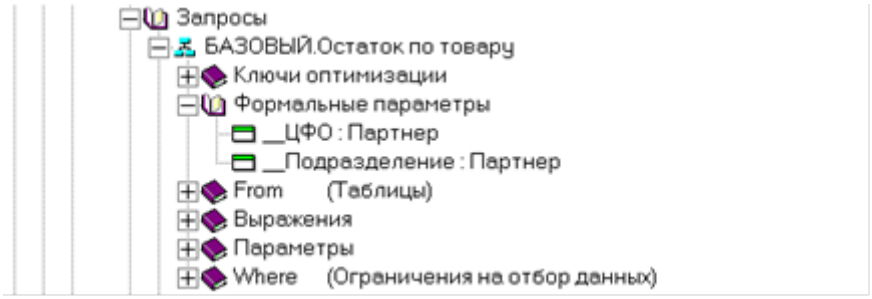


- **<Наследуемые пункты>** - этот атрибут применяется для наследования атрибутов раздела.
- **Быстрый показ первых записей (FIRST_ROWS)** – данный признак применяется в видах просмотра. Наличие признака означает, что не следует дожидаться получения всего результата запроса. Для вида просмотра достаточно получить лишь то количество записей, которое помещается на одном экране.
- **Порядок соединения таблиц брать из раздела FROM (ORDERED) –** данный признак используется только при работе с СУБД Oracle. При установленном признаке в запрос добавляется ключ оптимизации /* + ORDERED */. Данный ключ указывает, что соединение таблиц необходимо выполнить именно в том порядке, в котором таблицы перечислены в разделе FROM.
- **Использовать индекс,**
- **Использовать прямой порядок индекса,**
- **Использовать обратный порядок индекса –** позволяют явно указать индекс, который следует применить для эффективного выполнения запроса. Использование индекса может существенно ускорить фильтрацию или сортировку результатов запроса.

Формальные параметры

Данный раздел содержит список формальных параметров для вызова запроса. В общем случае запрос можно рассматривать как функцию специального типа. И для запроса применима возможность задания формальных параметров.

Обычно, формальные параметры используются при описании сложного базового запроса с множеством ограничений. Через формальные параметры передаются значения этих ограничений. Также с помощью формальных параметров можно указать условия включения в запрос тех или иных частей. Такие условия проверяются программой на этапе формирования запроса.

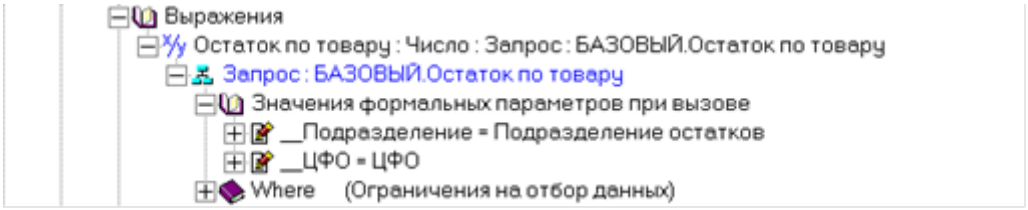


Запрос имеет два формальных параметра: __ЦФО и __Подразделение.

Значения формальных параметров при вызове

Данный раздел содержит значения формальных параметров при вызове запроса.

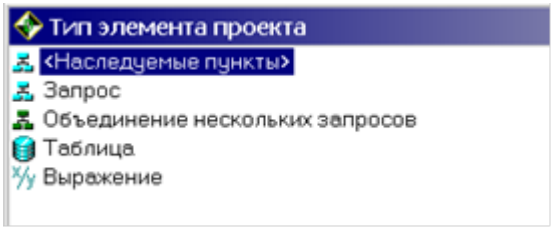
Использование данного раздела полностью аналогично его использованию в функциях и процедурах. Допустимо применение для вложенных запросов.



При вызове запроса, описанного в предыдущем примере, заполняются оба формальных параметра.

From (Таблицы)

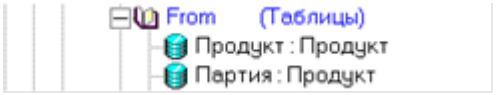
В данном разделе перечисляются источники, из которых отбираются данные с помощью запроса.



Можно указать одну или несколько таблиц.

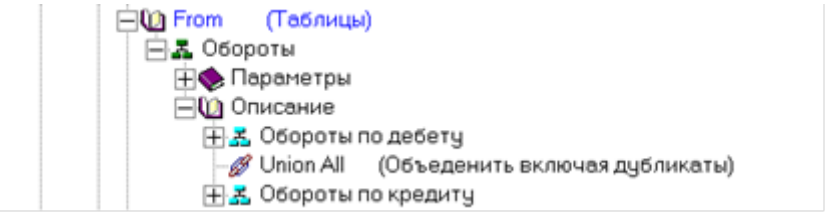
Запрос выполняется по соединению таблиц.

Если применяется соединение копий одной таблицы, то необходимо указать синонимы (поясняющие имена) для копий таблицы.



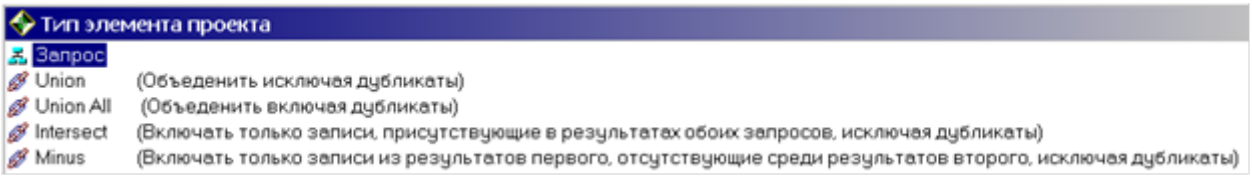
Запрос выполняется по соединению двух копий таблицы ‘Продукт’.

Можно указать вложенный запрос или объединение нескольких вложенных запросов.



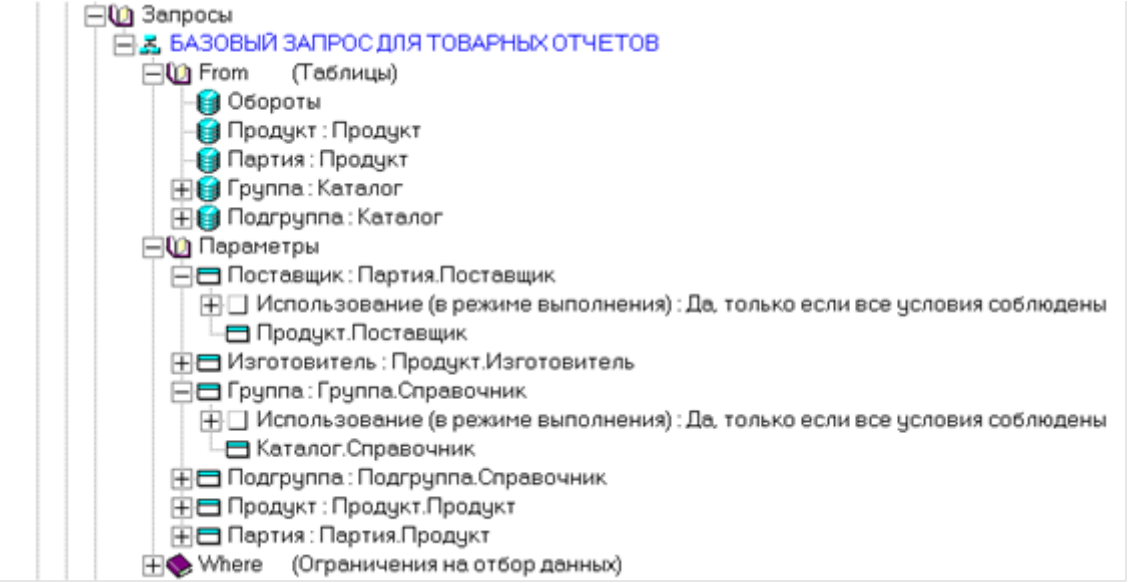
Запрос выполняется по результату вложенного объединенного запроса.

Допустимы следующие условия объединения запросов:

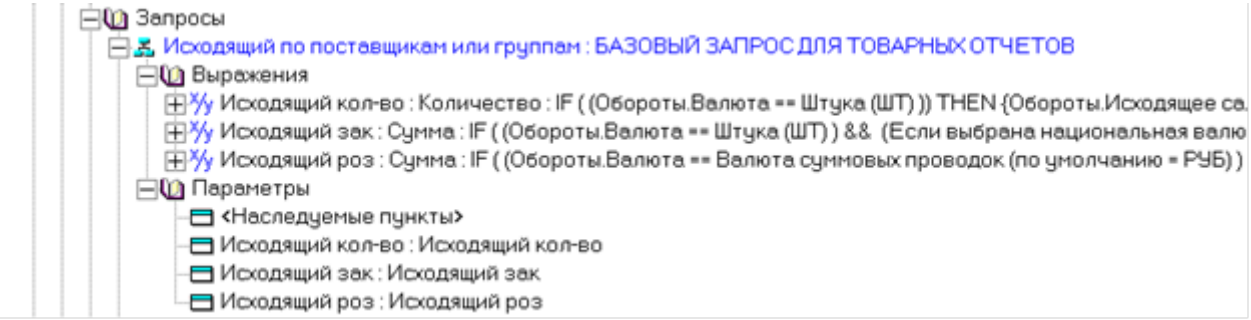


Параметры

Данный раздел содержит описание результата запроса. В результат запроса могут войти либо столбы таблицы, либо выражения с этими столбцами.



Базовый запрос в качестве результата вернет список значений следующих столбцов (перед наименованием столбца следует имя таблицы): ‘Партия.Поставщик’, ‘Продукт.Изготовитель’, ‘Группа.Справочник’, ‘Подгруппа.Справочник’, ‘Продукт.Продукт’, Партия.Продукт. Отдельно следует отметить, что запрос выполняется по трем таблицам БД: Обороты, Продукт и Каталог. Для соединения двух копий одной таблицы применяются синонимы. Для таблицы ‘Продукт’ описаны синонимы ‘Продукт’ и ‘Партия’. Для таблицы ‘Каталог’ описаны синонимы ‘Группа’ и ‘Подгруппа’. Наличие элемента 'Использование (в режиме выполнения)' означает, что в зависимости от описанного под этим элементом условия параметр может быть как включен в запрос, так и исключен из него.



Данный пример показывает, что результатом запроса могут быть не только поля таблиц, но и выражения. В результат запроса-наследника к столбцам, указанным в базовом запросе, добавляются три выражения. Выражения задаются в одноименном разделе.

Выражения

Данный раздел содержит описания выражений, которые могут использоваться в различных частях запроса (его результате, разделе WHERE и т.д.).

Пример находится выше.

Where (Ограничения на отбор данных)

В данном разделе описывается условие, в соответствии с которым записи включаются в результат запроса. Условие проверяется для каждой строки таблицы. Только те строки, для которых условие имеет истинное значение, попадают в результат.

Для описания условия можно использовать следующие атрибуты:

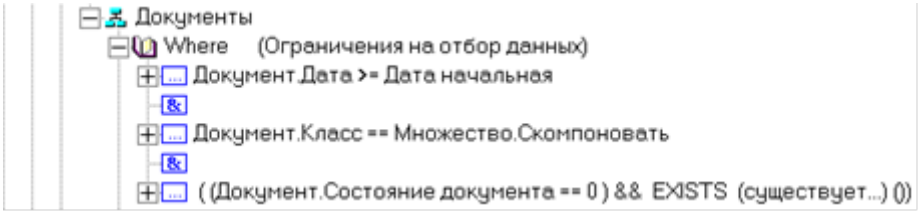


Большая часть из этих атрибутов похожа на соответствующие атрибуты выражения, но имеются существенные отличия. Если предназначение обычного выражения заключается в расчете выражения и возврате результата, то для описания запроса это не так.

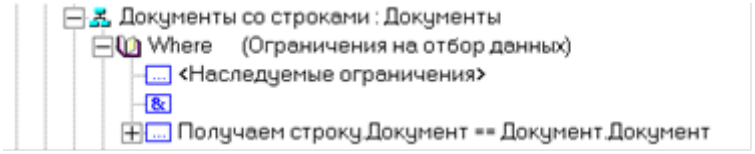
Описание запроса (в том числе и условие) должно быть преобразовано в конструкцию на языке SQL (на этапе формирования запроса), которая будет передана серверу БД для выполнения.

При описании условий запроса надо четко разделять - какие из них будут выполняться на сервере БД, а какие выполнятся на рабочей станции в момент формирования запроса. Сервер БД ничего не знает о ДОМИНО и потому формальные параметры, переменные и функции ДОМИНО могут быть для SQL запроса только константами, значение которых вычисляется на момент формирования запроса. Для ссылки на них используются атрибуты, начинающиеся со слова **Константа**. Для ссылки на поля таблиц, функции SQL сервера и выражения запроса применяются атрибуты, начинающиеся со слова **Результат**.

- **<Наследуемые ограничения>** - В задаваемое условие можно добавить условие из запроса-родителя. При добавлении условия используются булевы операции И, ИЛИ, НЕТ.



Это условие из запроса-родителя

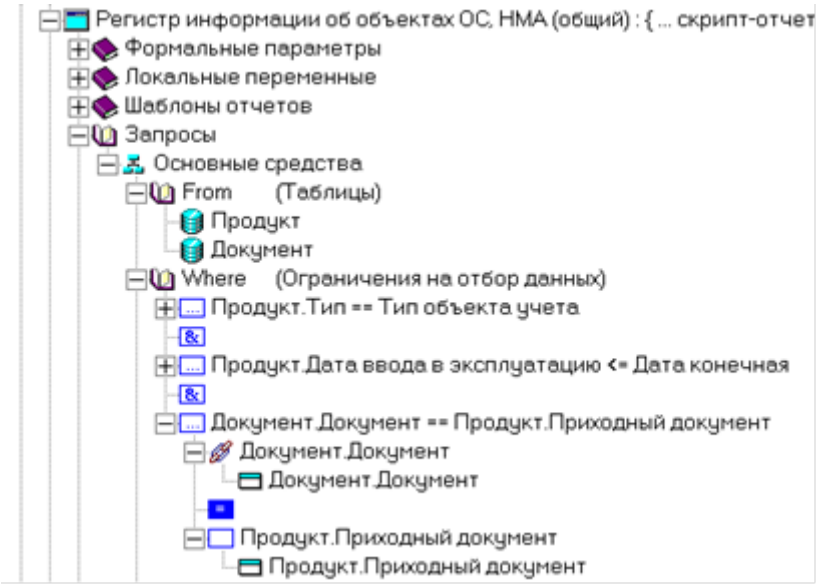


В запросе-наследнике к условию из запроса-родителя добавляется (операция И) еще одно условие.

- **Результат –** результат записывается в формируемый запрос без предварительных вычислений. Разрешено ссылаться только на объекты сервера БД (поля таблиц, функции SQL сервера, выражения запроса). На переменные, функции, выражения ДОМИНО ссылки запрещены (поскольку они неизвестны серверу БД).
- **Результат (Пустой UID, если NULL)** – к предыдущей конструкции добавляется дополнительная обработка результата (если результат имеет значение NULL, то возвращается пустой UID). Результат такого вида нужен при работе с таблицами ‘Права’ и ‘Сальдо’, ‘Нумераторы’, поскольку в этих таблицах вместо NULL значений хранятся пустые UID-ы.

В ORACLE запись пустого UID-а выглядит вот так – HEXTORAW ('00000000000000000').

- **Результат (Внешнее соединение таблиц – допустимо отсутствие значения)** – применяется для соединения таблиц. Для поля на которое ссылается данный операнд NULL значения считаются удовлетворяющими условию соединения таблиц.



В запросе соединяются таблицы ‘Продукт’ и ‘Документ’.

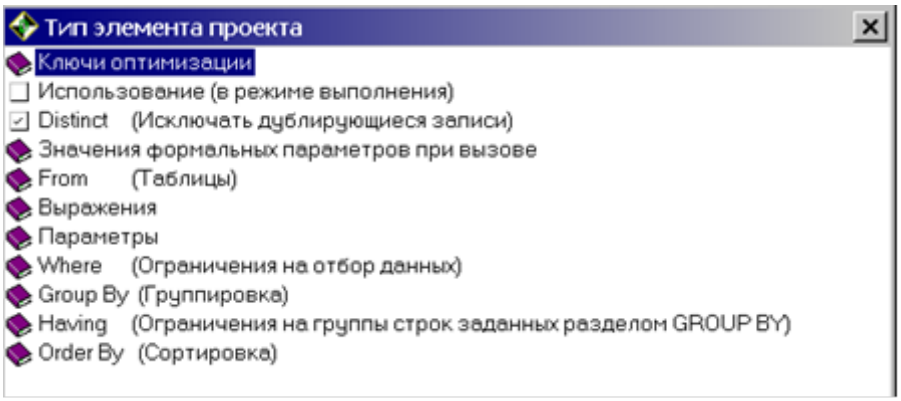
- **Константа** – атрибут будет записан в запрос как константа. Разрешено ссылаться на константы, переменные, функции и выражения ДОМИНО, значения которых будут вычислены при формировании запроса и добавлены в SQL конструкцию в качестве констант. Ссылки на объекты сервера БД (поля таблиц, функции SQL сервера, выражения запроса) недопустимы.
- **Константа (Ноль (0,0), если NULL)** – если константа имеет значение NULL, то возвращается ноль.
- **Константа (Пустой UID, если NULL)** – если константа имеет значение NULL, то возвращается пустой UID (HEXTORAW ('00000000000000000')).
- **Константа (Выражение) –** атрибут задает выражение, результат вычисления которого на этапе формирования запроса будет записан в SQL конструкцию как константа.
- **Константа (Ограничение по классу, типу, плану счетов)** –Появление данного атрибута связано с тем, что классы и типы объектов, в отличие от других проектных элементов, не записываются в БД целиком (как UID). Для классов и типов хранится только индекс проектного элемента. Применение данной константы позволяет напрямую ссылаться на проектный элемент, описывающий класс, тип, план счетов. Можно указать либо проектный элемент, либо выражение, возвращающее UID нужного проектного элемента.

Если атрибуты, начинающиеся со слова '**Константа**', возвращают результат типа UIDSET (список уникальных идентификаторов), то на этапе формирования запроса условие преобразуется либо в предложение IN на языке SQL, либо в предложение ANY. Предложение IN будет применено в том случае, если данная константа входит в выражение с операциями равно и не равно. Для остальных операций сравнения (больше, меньше и т.п.) будет использовано предложение ANY.

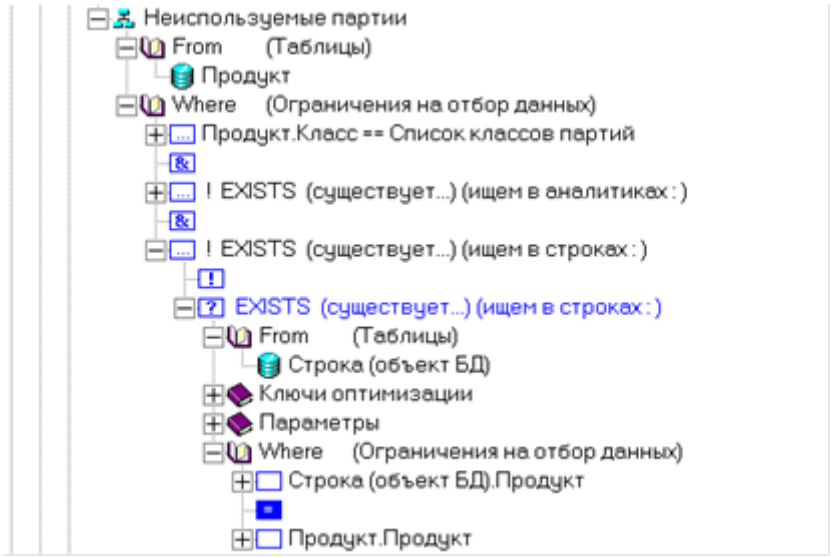
- **EXISTS (существует...) –** применяется для проверки факта наличия выходных

данных подзапроса. Оператор EXISTS возвращает либо истинное, либо ложное значение. Если подзапрос генерирует выходные данные, то оператор возвращает истинное значение. Ложное значение возвращается в противном случае.

Атрибуты для описания подзапроса внутри оператора EXISTS почти полностью совпадают с атрибутами запроса.



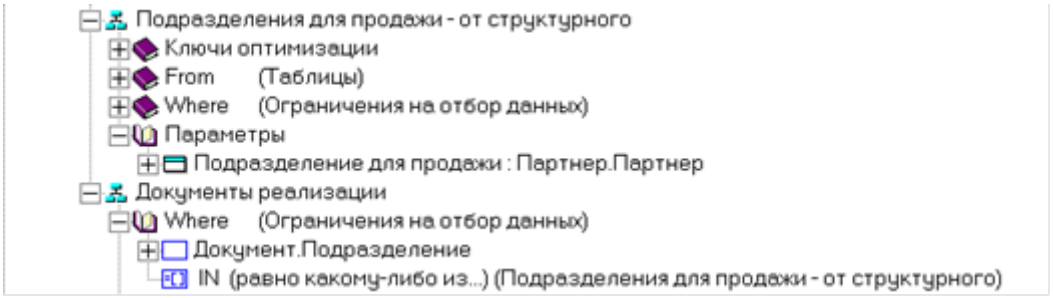
Рассмотрим пример использования оператора EXISTS.



При поиске неиспользуемых партий проверяется наличие партии в строках документов и аналитиках.

- **IN (равно какому-либо из ...)** – определяет множество, которому может принадлежать указанное значение. Множество значений генерируется подзапросом.

  Атрибуты для описания подзапроса внутри оператора IN почти полностью совпадают с атрибутами запроса.

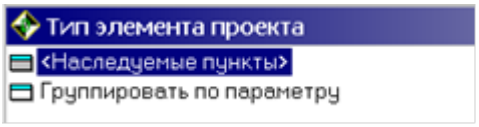


Первый запрос возвращает список подразделений. В условии второго запроса используется оператор IN, ссылающийся на результат первого запроса. Условие будет истинным в том случае, если подразделение из документа встретится среди подразделений, выбранных первым запросом.

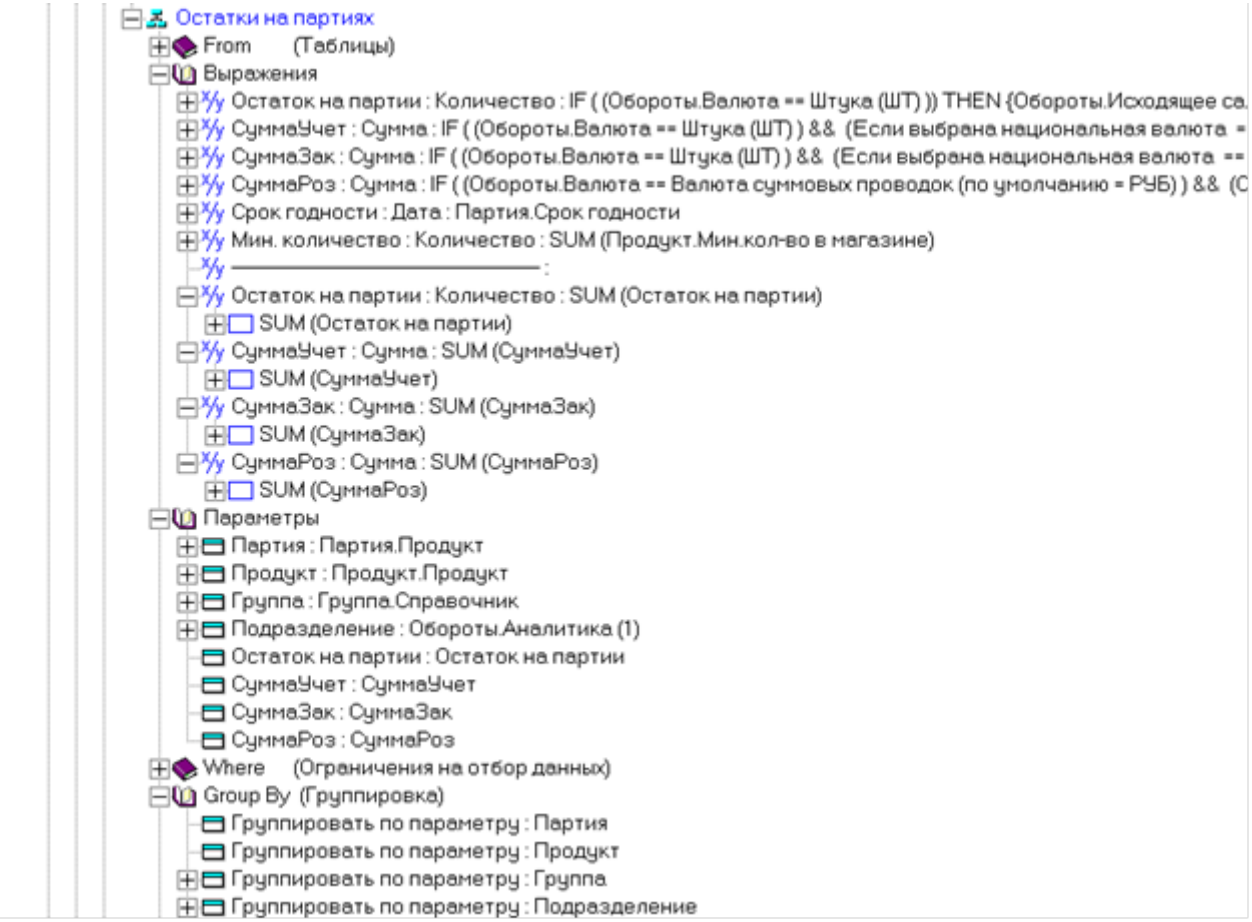
- **Запрос** – применяется для описания вложенного подзапроса. При использовании в качестве операнда выражения должен возвращать единственное значение.

Group By (Группировка)

В данном разделе описываются правила группировки записей для результата запроса. Наличие группировки позволяет применять к группам записей агрегатные функции сервера БД (COUNT, SUM, AVG, MAX, MIN).



Сначала записи будут сгруппированы по значению первого параметра. Внутри каждой из полученных групп записи будут сгруппированы по значению второго параметра. И так далее по всем указанным в разделе параметрам.



Каждая запись результата запроса содержит восемь полей (‘Партия’, ‘Продукт’, ‘Группа’, ‘Подразделение’, ‘Остаток’, три суммы). Записи будут сгруппированы сначала по значению поля ‘Партия’. Внутри полученных групп – по значению поля ‘Продукт’. Затем – по значению поля ‘Группа’. И последняя группировка – по значению поля ‘Подразделение’. В четырех выражениях вызывается функция агрегирования SUM, подсчитывающая арифметическую сумму значений указанных полей. Расчет будет выполнен для каждой группы записей.

- **<Наследуемые пункты>** - этот атрибут применяется для наследования группировки запроса-родителя.

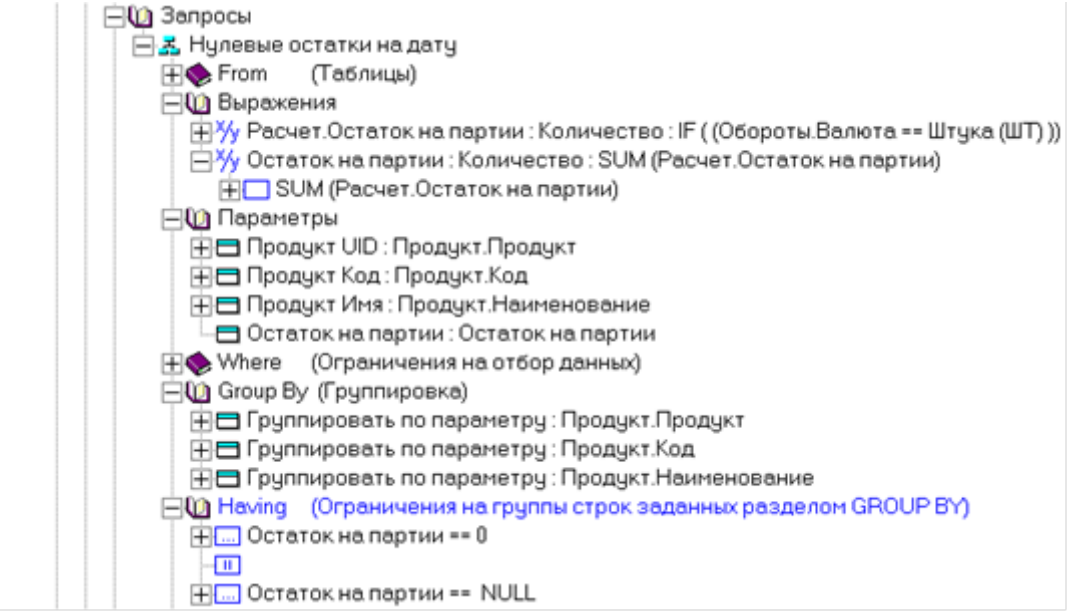
При использовании группировки в результат запроса можно включать только поля группировки и результаты агрегирования. Все прочие поля в результате запроса перечислены быть не могут.

Having (Ограничения на группы строк заданных разделом Group By)

В данном разделе описывается условие, которое применяется к группам, заданным в разделе GROUP BY. Группы, для которых указанное условие не выполняется, исключаются из выходных данных. Условие HAVING действует на группы строк, в отличие от условия WHERE, которое действует на одну строку. При описании условия HAVING можно применять

функции агрегирования.

Атрибуты для описания условие HAVING полностью совпадают с атрибутами для описания условия WHERE.



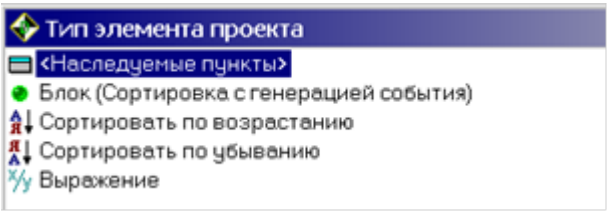
> *Запрос выдает продукты, имеющие нулевые остатки на указанную дату. В выходные данные попадают только те продукты, у которых ни для одной партии не найден остаток. Для расчета общего остатка по всем партиям применяется функция агрегирования SUM (ее вызов находится в выражении 'Остаток на партии').*

> При группировке результат запроса сжимается до полей группировки и результатов агрегирования и, следовательно, обращение к прочим полям в данном разделе невозможно.

Order By (Сортировка)

В данном разделе задаются правила сортировки выходных данных запроса. Записи упорядочиваются в соответствии со значениями одного или нескольких указанных полей. При указании нескольких полей записи сначала группируются и упорядочиваются по значению первого поля. Внутри групп записи сортируются по значению второго поля, и т.д. по всем указанным полям.

При описании сортировки применяются следующие атрибуты:



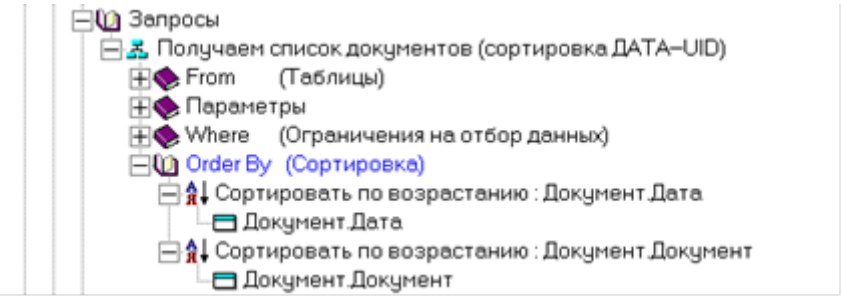
- **<Наследуемые пункты>** - этот атрибут применяется для наследования сортировок запроса-родителя.
- **Блок (Сортировка с генерацией события)** – применяется для группировки отсортированных записей. В блоке объединяются записи с одинаковыми значениями тех ключевых полей, описание которых находится внутри блока. Последовательность расположения блоков определяет уровень (вложенность) группировки записей.

  При отборе данных отслеживается изменение ключевых полей. Как только поменяется условие группировки блока (значения ключевых полей блока), программа для первой записи с новыми значениями ключевых полей генерирует событие *‘Выполнить перед обработкой блока’*, а после обработки последней записи с таким же значением ключа – событие ‘*Выполнить после обработки блока*’.

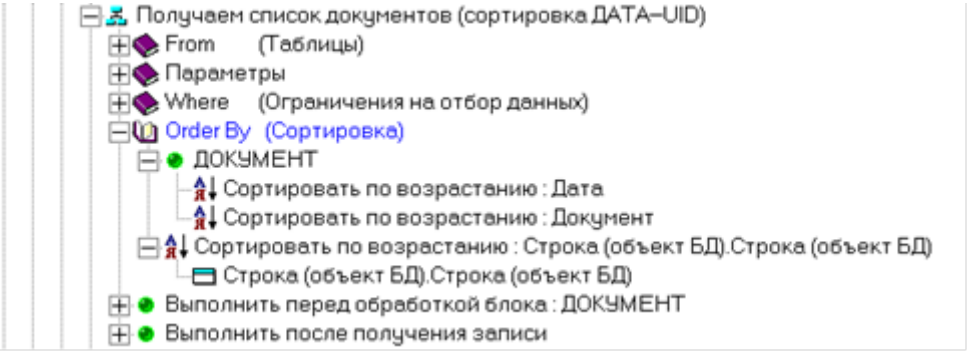
  Самый нижний уровень группировки (поля вне блоков) попадает в одно событие '*Выполнить после получения записи*', которое генерируется для каждой новой записи при отборе данных (после события ‘*Выполнить перед обработкой блока*’, но перед событием ‘*Выполнить после обработки блока*’).

  Для каждого поля можно указать вид сортировки по этому полю: по убыванию значения поля или по возрастанию значения.

- **Сортировать по возрастанию** – записи будут отсортированы по возрастанию значения указанного поля таблицы. Можно ссылаться на параметры запроса.
- **Сортировать по убыванию** – записи будут отсортированы по убыванию значения указанного поля таблицы. Можно ссылаться на параметры запроса.
- **Выражение** – выражения из этой папки применяются при описании других атрибутов ORDER BY.



Список документов сортируется по дате документа, внутри одинаковых дат – по UID-у документа.



Список документов со строками сортируется по дате документа, UID-у документа, UID-у строки. В запросе имеется обработка двух событий. Первое событие генерируется при изменении документа (дата и UID). Второе событие генерируется при отборе очередной записи.

Версия #3

Демонов Сергей создал Thu, Mar 31, 2022 6:05 PM

Демонов Сергей обновил Thu, Mar 31, 2022 6:52 PM