

Группа компаний "Информация и управление"

***Инструментальные средства объектной
разработки программных систем
S_Технология (SDK S_Технология)***

**Руководство по модулю
*Генератор отчетов***

Редакция 2.0.0.X

Воронеж, 2018

АННОТАЦИЯ

Настоящая инструкция содержит описание модуля «Генератор отчетов» инструментальных средств объектной разработки программных систем *S_Технология*. Даны определения основных понятий и приведено описание основных функций и интерфейса модуля.

Инструментальные средства объектной разработки программных систем *S_Технология* (SDK *S_Технология*) разработаны группой компаний "Информация и управление" (компания "ИнфоМега" и "Информация и управление"), г. Воронеж.

Документ соответствует версии SDK *S_Технология*, начиная с редакции 2.0.0.X, до выпуска новой редакции.

СОДЕРЖАНИЕ

ТЕРМИНЫ И СОКРАЩЕНИЯ	5
ВВЕДЕНИЕ	5
НАЗНАЧЕНИЕ И ОБЩИЙ ПРИНЦИП РАБОТЫ ГЕНЕРАТОРА ОТЧЕТОВ	5
ОПИСАНИЕ И ЗАПУСК МОДУЛЯ	5
ФОРМА ГЕНЕРАТОРА ОТЧЕТОВ.....	6
ПАНЕЛЬ УПРАВЛЕНИЯ ОТЧЕТОМ	7
ПАНЕЛЬ ВИЗУАЛЬНОЙ ЧАСТИ ОТЧЕТА.....	7
ПАРАМЕТРЫ ОТЧЕТА	9
<i>Разновидности параметров отчета.....</i>	<i>11</i>
<i>Примеры создания параметров отчета</i>	<i>11</i>
ЦИКЛЫ ОТЧЕТА.....	13
<i>Описание цикла</i>	<i>13</i>
<i>Размер цикла</i>	<i>15</i>
<i>Отладка циклов.....</i>	<i>16</i>
<i>Дополнительные выражения цикла</i>	<i>16</i>
<i>Системные переменные, связанные с циклами</i>	<i>17</i>
<i>Горизонтальные и вертикальные циклы</i>	<i>17</i>
<i>Пересечения циклов. Специальные параметры и переменные</i>	<i>17</i>
СВОЙСТВА ПАРАМЕТРОВ И ЦИКЛОВ ОТЧЕТА.....	18
ВЫРАЖЕНИЯ ОТЧЕТА (ФОРМУЛЫ).....	19
<i>Имена выражений. Ссылки на выражения</i>	<i>20</i>
<i>Переменные выражений</i>	<i>22</i>
<i>Выражения и их значения</i>	<i>23</i>
ФОРМА ВВОДА ФОРМУЛ.....	23
ОПРЕДЕЛЕНИЕ СУЩНОСТИ	27
ВВОД ФОРМУЛЫ.....	29
ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА.....	33
АГРЕГАТНЫЕ ФУНКЦИИ.....	33
ВСТРОЕННЫЕ ФУНКЦИИ	34
<i>Математические функции</i>	<i>34</i>
<i>Логические функции.....</i>	<i>35</i>
<i>Строковые функции</i>	<i>36</i>
<i>Функции форматирования.....</i>	<i>37</i>
<i>Функции даты и времени</i>	<i>39</i>
<i>Групповые функции.....</i>	<i>39</i>
<i>Агрегатные функции SQL.....</i>	<i>39</i>
<i>Списочные функции</i>	<i>40</i>
<i>Другие функции</i>	<i>42</i>
СКРИПТЫ	44
ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ.....	44
ДОПОЛНИТЕЛЬНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА	45
ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ ПРИ ПОЛУЧЕНИИ СВОЙСТВА СУЩНОСТИ	45
<i>Параметр «Фильтр».....</i>	<i>46</i>
<i>Параметр «Цикл».....</i>	<i>46</i>
<i>Параметр «Запись».....</i>	<i>47</i>
<i>Параметр «Сортировка»</i>	<i>47</i>
<i>Параметр «Привязка»</i>	<i>48</i>
<i>Параметр «Фильтр_списка»</i>	<i>49</i>

<i>Параметр «Пустое_Поле»</i>	50
<i>Параметр «Формат_даты»</i>	50
<i>Параметр «Группировка»</i>	51
СИСТЕМНЫЕ СУЩНОСТИ	51
СИСТЕМНЫЕ ФУНКЦИИ	52
ЗАВЕРШАЮЩИЙ МАКРОС	54
СПЕЦИАЛЬНЫЕ СИМВОЛЫ И ЭФФЕКТЫ	54
СОЗДАНИЕ НОВОГО ОТЧЕТА	55
СОЗДАНИЕ ШАБЛОНА	55
СОХРАНЕНИЕ ШАБЛОНА	55
ОБЩИЙ ПОРЯДОК СОЗДАНИЯ ОТЧЕТА.....	55
ФОРМИРОВАНИЕ И ПРОСМОТР ОТЧЕТА	56
ПРИЛОЖЕНИЕ 1. ПРИВИЛЕГИИ ДЛЯ РАБОТЫ С МОДУЛЕМ «ГЕНЕРАТОР ОТЧЕТОВ»	57
ПРИЛОЖЕНИЕ 2. ДЕЙСТВИЯ ПО КЛАВИШАМ	57

Термины и сокращения

БД	База данных
Шаблон отчета	Пара файлов с одинаковым наименованием, но разного формата – <i>xls</i> и <i>rpt</i> , которые хранят настройку отчета – внешний вид и формулы для обращения к БД с целью получения данных для вывода в отчет.
Отчет	Файл MS Excel, сформированный по заранее подготовленному <i>шаблону отчета</i> для заданных параметров.
Параметр отчета	Специальная переменная, предусмотренная <i>шаблоном</i> , значение которой выбирается непосредственно перед запуском отчета на выполнение; от выбранного значения зависит конечный результат. Выбирая разные значения <i>параметров</i> , можно получать разные <i>отчеты</i> .
Цикл отчета	Область данных в <i>шаблоне</i> , которая при выводе <i>отчета</i> повторяется некоторое число раз.
Выражение отчета	Формулы, с помощью которых выполняются различные операции над данными, полученными из БД, необходимые для формирования <i>отчета</i> .
ИМ	Группа компаний «Информация и управление».

Введение

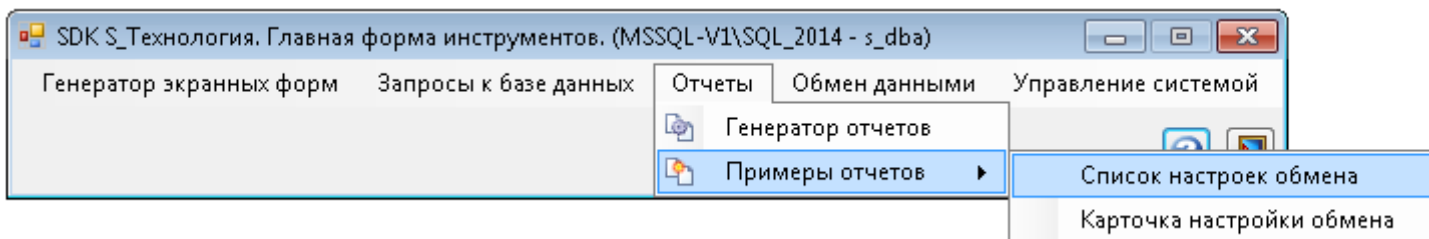
Инструментальные средства объектной разработки программных систем *S_Технология* предназначены для разработки информационных систем любой сложности на платформе .Net в среде разработки Microsoft Visual Studio на языке C# для архитектур Windows Forms и Web.

Назначение и общий принцип работы Генератора отчетов

Модуль «Генератор отчетов» предназначен для подготовки шаблонов отчетов в MS Excel, которые могут быть подключены к проекту на базе *S_Технологии*. Принцип работы состоит в том, что пользователь готовит шаблон отчета, который состоит из визуальной части (вид отчета в MS Excel) и формул, с помощью которых можно обращаться к базе данных, выполнять различные вычисления и выводить результат в отчет. Готовый шаблон отчета можно запустить на выполнение как из среды генератора отчетов, так и из любого места системы. Можно настроить много различных вариантов шаблонов, исходя из особенностей конкретной системы.

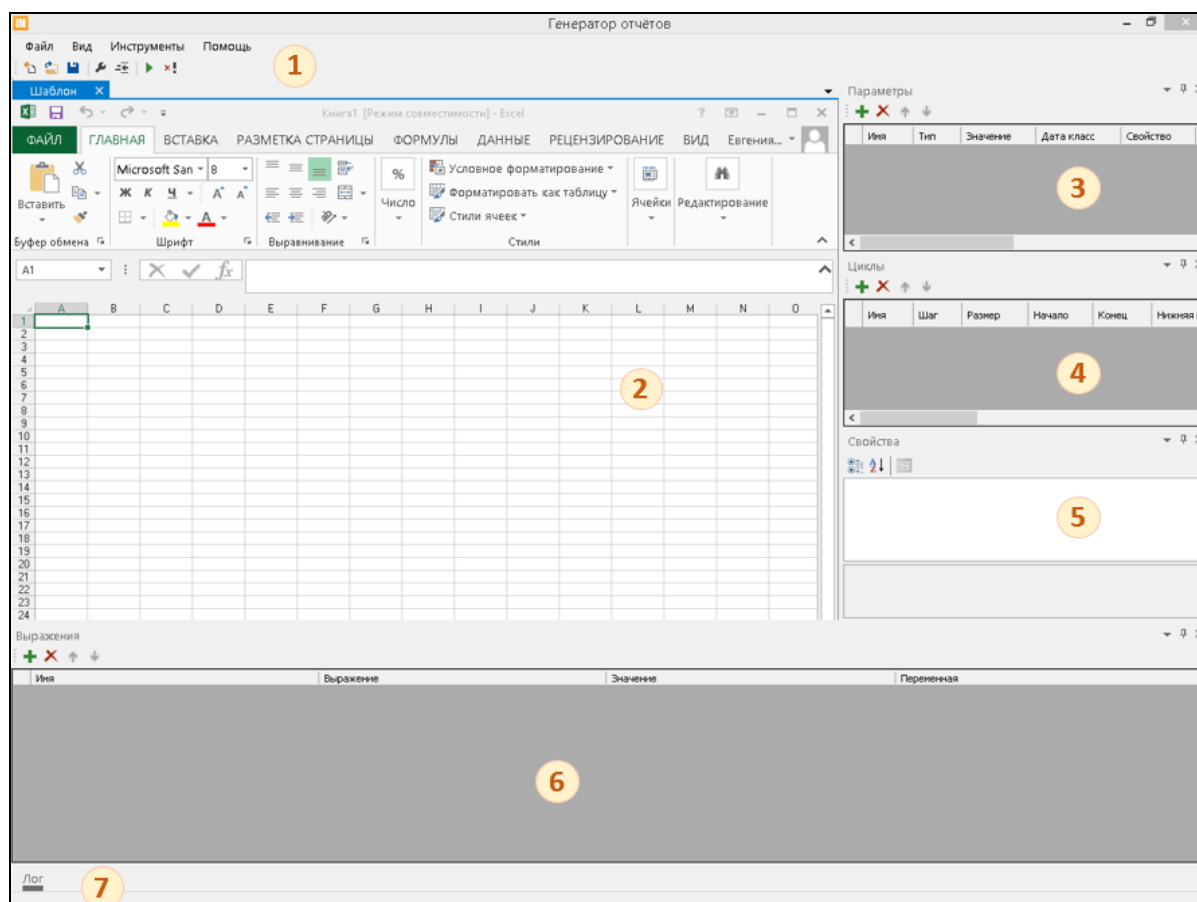
Описание и запуск модуля

Модуль представлен формой [Генератор отчетов](#) и программой, запускающей отчеты на выполнение. Вызывается по кнопке *Генератор отчетов* раздела «Отчеты» главной формы инструментов. Ниже приведен скриншот, показывающий вызов модуля на главной форме инструментов (на примере комплекта поставки Extended).



Форма Генератора отчетов

Форма «Генератор отчетов» предназначена для просмотра и редактирования шаблонов отчетов, а также для запуска отчетов по существующим или вновь созданным шаблонам. Вид формы «Генератор отчетов» представлен на рисунке.



Форма состоит из следующих основных элементов.

1. Панель управления отчётом.
2. Панель визуальной части отчёта (встроенный MS Excel).
3. Список параметров отчёта.
4. Список циклов отчёта.
5. Панель свойств текущего параметра/цикла.
6. Панель выражений отчёта.
7. Лог отчета.

Ниже в отдельных разделах приведено описание этих элементов и их использование для редактирования шаблонов отчетов.

Панель управления отчетом

Панель управления отчетом содержит меню управления и кнопки быстрого доступа к некоторым функциям. Вид панели приведен на рисунке ниже.



Меню *Файл* содержит пункты для управления файлами отчетов (пункты продублированы кнопками быстрого доступа).

- *Новый*. Выполняется создание нового шаблона.
- *Открыть...* Выполняется загрузка существующего шаблона.
- *Сохранить*. Выполняется сохранение изменений в текущем шаблоне.

Меню *Вид* содержит пункты, которые позволяют управлять видом формы генератора отчетов – переключаться на *Шаблон*, открывать и закрывать вкладки *Свойств* параметров и циклов, *Лога*, *Строки состояния* и др.

Меню *Инструменты* содержит пункты для выбора цветовой схемы формы. Здесь же можно выставить флаг *Полный пересчет*, который указывает правила расчета выражений отчета (после ввода очередного выражения, будут ли вычисляться все выражения, либо только текущее и последующие выражения).

Меню *Помощь* предназначено для вызова справки.

Также на панели управления представлены кнопки, не продублированные в меню.

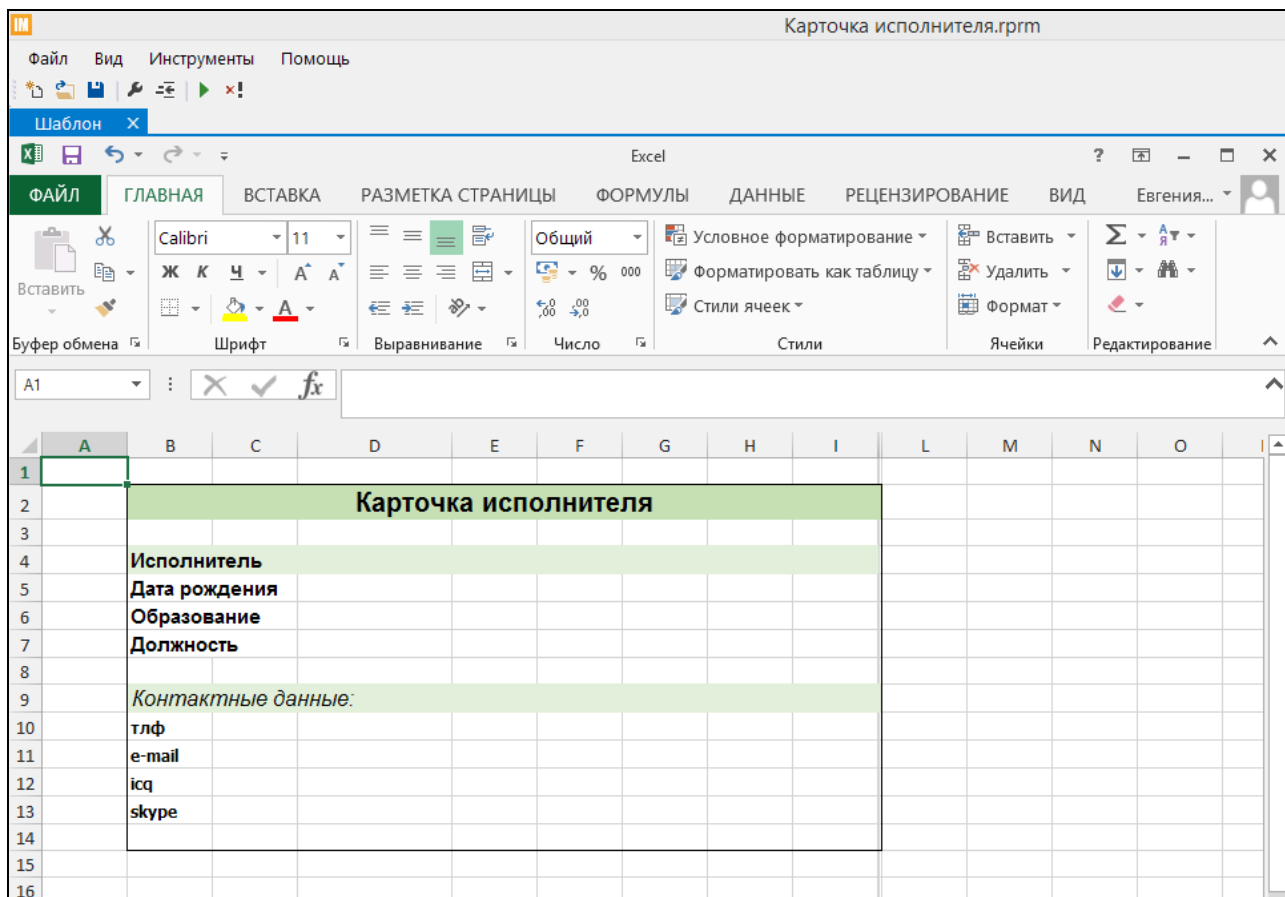
- *Запуск*. Формирование отчета по текущему шаблону.
- *Очистить буфер*. Выполняется очистка буфера временных переменных и пересчет всех выражений отчёта.

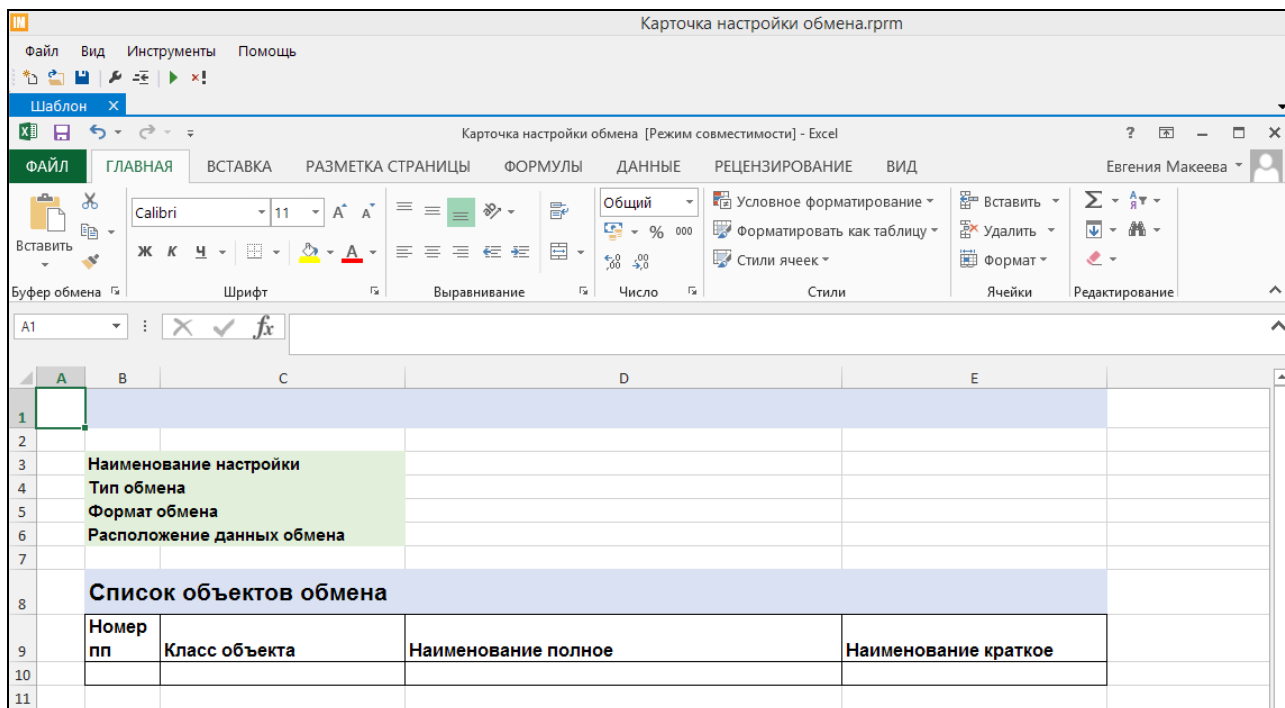
Панель визуальной части отчета

Панель визуальной части отчета представляет собой встроенную форму MS Office Excel. Подробнее см. в документации или справке Microsoft Excel.

На этой панели средствами Excel может быть настроен любой «шаблон», в которой впоследствии будут размещаться данные, рассчитанные по формулам, заданным в отчете. Можно добавить заголовок отчета, наименования выводимых полей, задать шрифт, размер, цвет и другие параметры ячеек, добавить таблицы, границы и т.д.

Ниже приведено несколько примеров настройки шаблонов Excel.





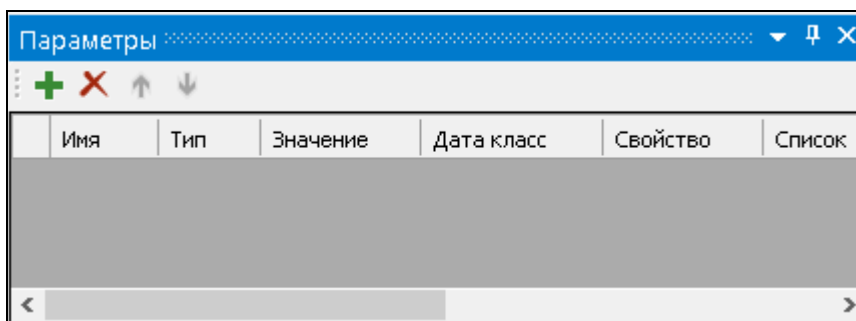
Параметры отчета

Параметры – переменные от которых зависит расчёт выражений в отчёте. Параметры могу использоваться:

- при создании условий на выборку данных;
- самостоятельными источниками данных (например, быть списком или перечислением);
- в логических выражения (скрипты, функции, простые выражения).

Значения параметров могут задаваться непосредственно в генераторе отчётов (в процессе написания выражений для отладки новых шаблонов). Для запуска уже готовых шаблонов отчетов (не из среды Генератора отчетов) значения параметров могут задаваться либо программистом в runtime, либо пользователем в интерактивном режиме запуска отчёта (когда перед запуском у пользователя запрашиваются значения параметров).

Панель «Параметры» состоит из списка параметров и кнопок управления.



Список колонок на сетке (списке) параметров.

- *Имя* – наименование параметра. Обязательное поле. Имена параметров обязаны начинаться с символа «:» и могут состоять из произвольного набора латинских и русских букв и цифр, разделенных знаком подчеркивания; пробелы и другие символы не допускаются.

- *Тип* – тип параметра. Обязательное поле. Возможные значения: целое, вещественное, строка, логическое, дата, время.
- *Значение* – текущее значение параметра. Обязательное поле для обязательных параметров, для необязательных значение может отсутствовать.
- *Дата класс* – имя класса, поле которого является параметром. Необязательное поле.
- *Свойство* – имя атрибута дата класса, который является параметром. Необязательное поле, имеет смысл только при непустом значении *Дата класса*.
- *Список* – позволяет указать, что параметр является списочным (по умолчанию он таковым не является). Если параметр задан как списочный, это означает, что пользователь может выбрать сразу несколько значений из связанной с параметром сущности. Тип параметра в этом случае всегда будет «Строка», а значением параметра будет являться список из значений свойств для всех выбранных записей, перечисленных через символ «;».
- *Обязательный* – позволяет указать, что значение параметра при запуске отчета не может быть пустым и должно быть обязательно заполнено пользователем. По умолчанию все параметры являются обязательными; если для какого-то из них допустимо не указывать значение, признак обязательности для него должен быть снят.
- *Перечисление* – с помощью этого параметра можно указать набор строк, которые будут предоставлены пользователю для выбора из них значения параметра. Значения строк разделяются символом «;», при необходимости указать в значении символ «;» все значение заключается в кавычки, при необходимости в этом случае указать внутри значения кавычку, она удваивается. Если такой параметр с перечислением имеет тип строки, то его значением после выбора будет сама выбранная строка, если же он имеет тип целого или вещественного числа, то значением параметра будет порядковый номер выбранной строки (начиная с 1).
- *Зависимость* – позволяет указать зависимость данного параметра от других параметров. Для указания зависимости перечисляются порядковые номера параметров (начиная с 1), номера отделяются друг от друга символом «;». Зависимость от параметра означает, что при изменении значения одного из параметров, от которых зависит текущий параметр, значение текущего параметра очищается. Кроме того, обычно зависимость от параметра означает, что список значений параметра, из которого делается выбор, зависит от значения зависимого параметра (хотя это и не обязательно). Данная зависимость описывается в виде выражения связи, которое задается в столбце «Связь».
- *Связь* – позволяет указать выражение для связи значений параметра, предоставляемых для выбора пользователю, с значениями других параметров отчета. Выражение для связи должно представлять собой логическое выражение, в котором могут использоваться имена полей в сущности, от которой зависит параметр, а также ссылки на значения других параметров отчета. Эти ссылки имеют вид «*\$\$PARAM<n>*», где *n* – порядковый номер параметра в отчете (*\$\$PARAM1*, *\$\$PARAM3* и т.д.).

Список кнопок управления параметрами.

- *Добавить строку*. Кнопка добавляет новый параметр в текущий шаблон.
- *Удалить строку*. Кнопка удаляет текущий параметр из текущего шаблона.
- *Переместить вверх*, *Переместить вниз*. Кнопки предназначены для управления порядком следования параметров в шаблоне.

Разновидности параметров отчета

В отчетах могут встречаться две разновидности параметров. Первая – это какие-либо независимые фиксированные значения – даты, числа, строки и т.д. Например, это может быть начальная дата какого-либо периода, минимальное количество записей для включения строки данных в отчет и прочие аналогичные значения. Эти данные не являются данными из базы. Для таких параметров необходимо указывать имя и тип данных параметра.

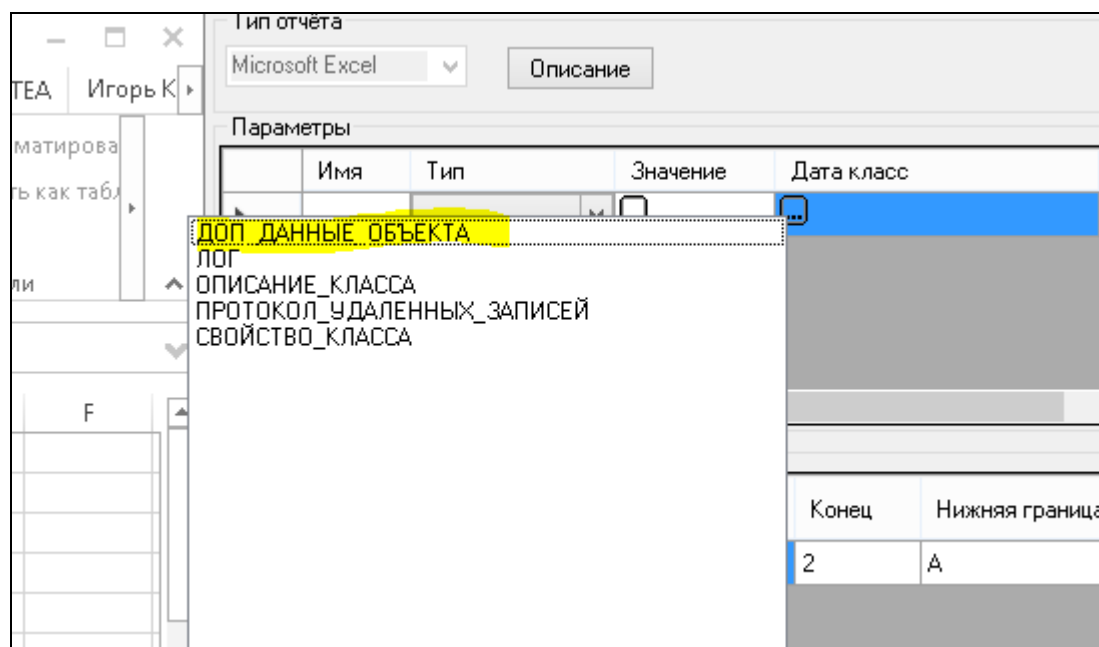
Другой разновидностью являются параметры, значения которых берутся непосредственно из базы данных. Это может быть внутренний код (ID) для какой-то сущности, наименование какого-либо объекта, имя исполнителя и т.д. Для этих параметров можно явно указать сущность и свойство в ней, связанное с данным параметром. После этого, во-первых, становится определенным тип данных параметра (он совпадает с типом данных указанного свойства сущности), и во-вторых, значение параметра можно задавать не прямым вводом, а выбором необходимого значения из базы данных.

Ниже приведено несколько примеров создания параметров.

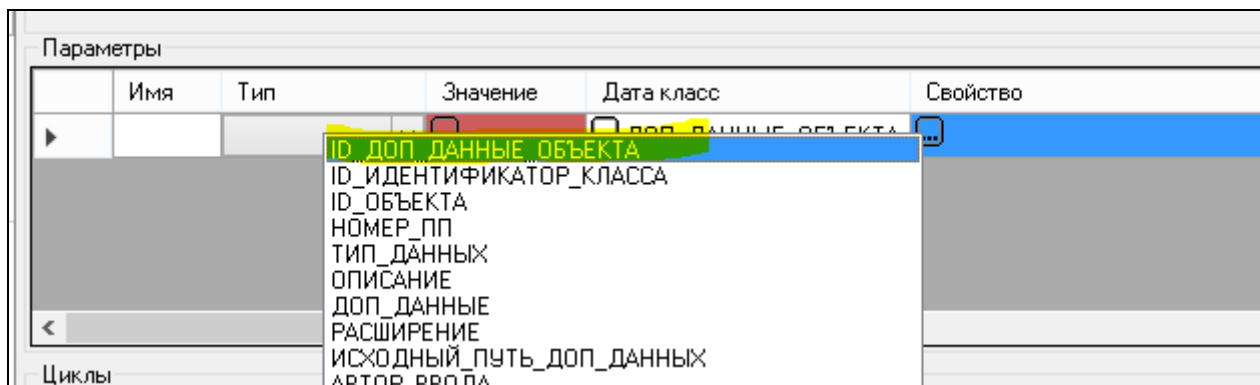
Примеры создания параметров отчета

Параметр – значение из таблицы БД. Наиболее используемый вариант параметра. Для создания такого параметра нужно сделать следующее.

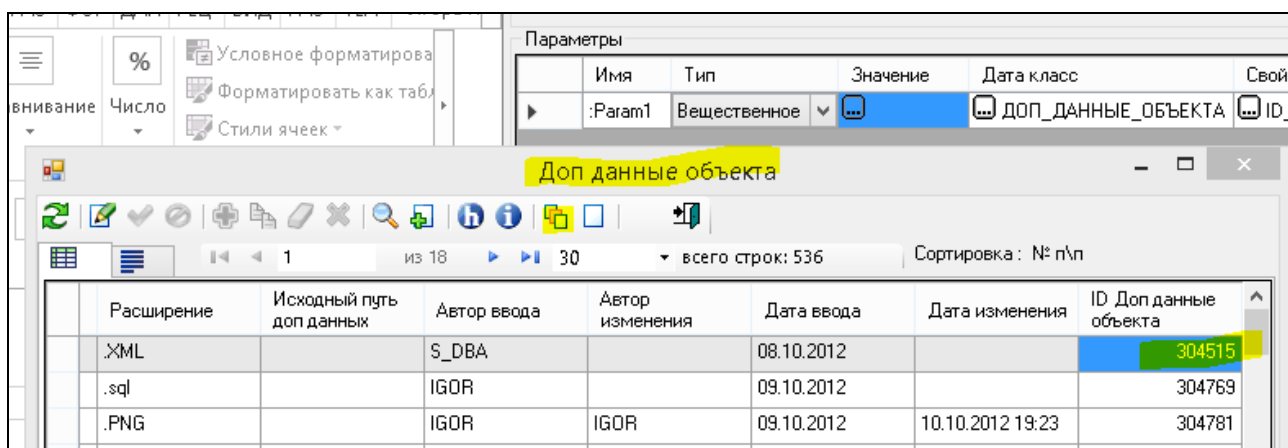
- Добавить запись нового параметра по кнопке *Добавить строку*.
- В колонках *Список* и *Обязательный* будут выставлены значения по умолчанию: False и True соответственно. Т.е. параметр не является списком значений и он обязательный.
- Задать *Имя* параметра, начинающееся с двоеточия, например, *:Param1*.
- В колонке *Дата класса* выбрать таблицу (сущность), поле которой будет параметром.



- В колонке *Свойство* выбрать свойство таблицы (сущности), которое будет параметром.



- Тип параметра в колонке *Тип* будет автоматически выставлен в зависимости от выбранного *Свойства*.
- В колонке *Значение* выбрать значение параметра из стандартной формы выбранной сущности.



После проделанных шагов, параметр готов к использованию в отчете. Значение ID_ДОП_ДАННЫЕ_ОБЪЕКТА будет доступно в переменной **:Param1**. Сетка параметров будет выглядеть следующим образом.

Параметры						
	Имя	Тип	Значение	Дата класс	Свойство	Список
	:Param1	Вещественное	304515	ДОП_ДАННЫЕ_ОБЪЕКТА	ID_ДОП_ДАННЫЕ_ОБЪЕКТА	False

Параметр – список значений из таблицы БД. Также часто используемый вариант параметра. Списочный параметр создается так же, как и параметр с единственным значением. Отличие состоит в том, что для списочного параметра в поле *Список* выбирается значение True.

Отдельно нужно сказать о некоторых особенностях данного типа параметров. Во-первых, тип этого параметра всегда должен быть «Строка». Во-вторых, значением параметра будут несколько значений, разделенных точкой с запятой (т.е. строка следующего формата 1;2;3...).

Параметры						
	Имя	Тип	Значение	Дата класс	Свойство	Список
▶	Param1	Строка	304515:3047...	ДОП_ДАННЫЕ_ОБЪЕКТА	ID_ДОП_ДАННЫЕ_ОБЪЕКТА	True

Параметр – константа (не из таблицы БД). Чтобы создать простой параметр, который является константой, нужно задать только *Имя*, *Тип* и *Значение* параметра, *Дата класс* и *Свойство* не задаются.

Параметры						
	Имя	Тип	Значение	Дата класс	Свойство	Список
▶	:Param1	Целое	5			

Простой тоже параметр может быть списочным. Для этого нужно выставить *Тип* в значение «строка», а *Список* в значение True, и ввести список значений через точку с запятой в поле *Значение*.

Циклы отчета

Достаточно часто в отчете бывает необходимо отобразить какую-либо информацию для каждой записи из некоторого набора. Например, напечатать список заданий или список исполнителей по проекту.

Для выполнения данной задачи в языке построения отчетов используется специальное понятие – цикл. Циклом называется некоторая прямоугольная область данных на листе Excel отчета, которая при выводе отчета повторяется некоторое число раз. Количество повторений зависит от формул, использующихся для заполнения этой области, и от текущих значений параметров.

Циклы задаются непосредственно в генераторе отчётов. Любой шаблон может иметь один или несколько циклов (необязательно). Панель «Циклы» состоит из списка циклов и кнопок управления.

Циклы							
+	×	↑	↓				
	Имя	Шаг	Размер	Начало	Конец	Нижняя граница	Верхняя грани

Описание цикла

Список колонок на сетке (списке) циклов.

- *Имя* – наименование цикла.
- *Шаг* – текущий шаг цикла.
- *Размер* – расчётный размер цикла (количество строк).
- *Начало* – начальная строка excel для того диапазона ячеек, который должен повторяться в цикле.

- *Конец* – конечная строка excel для того диапазона ячеек, который должен повторяться в цикле.
- *Нижняя граница* – начальный столбец excel для того диапазона ячеек, который должен повторяться в цикле.
- *Верхняя граница* – конечный столбец excel для того диапазона ячеек, который должен повторяться в цикле.
- *Сортировка* – выражения, которые должны участвовать в сортировке таблицы цикла.
- *Доп.выражения* – выражения, которые явно не выводятся в цикле, но должны быть пересчитаны на каждом его шаге.
- *Исключение для диаграмм (строки), Исключение для диаграмм(столбцы)* – выражения, которые используются для построения диаграмм в отчетах.

Список кнопок управления циклами.

- *Добавить строку*. Кнопка добавляет новый цикл в текущий шаблон.
- *Удалить строку*. Кнопка удаляет текущий цикл из текущего шаблона.
- *Переместить вверх, Переместить вниз*. Кнопки предназначены для управления порядком следования циклов в шаблоне.

Создаются циклы так же, как и параметры. Нужно нажать кнопку *Добавить строку* и заполнить ее. Для определения цикла необходимо указать его имя и задать область ячеек на листе отчета, которая будет повторяться для каждого шага цикла. Если точнее, копия указанной области ячеек будет для каждого шага цикла вставляться под этой самой областью. При этом ячейки, находящиеся ниже, будут сдвинуты вниз. При копировании сохраняются все атрибуты копируемых ячеек – размер и начертание шрифта, выравнивание, границы, цветовое оформление, а также высота соответствующих строк.

В колонку *Имя* вводится имя цикла. Оно может быть произвольным и состоять из русских и латинских букв и цифр. В основном оно несет только информативную функцию и практически никогда явно не используется.

В столбцы *Начало* и *Конец* вводятся начальная и конечная строки цикла. Как правило, чаще всего цикл состоит только из одной повторяющейся строки, и конечная строка совпадает с начальной. Но возможна и ситуация, когда под каждую запись отводится несколько последовательных строк. Тогда для каждого шага цикла будет копироваться вся группа строк (точнее, диапазон ячеек из нескольких строк).

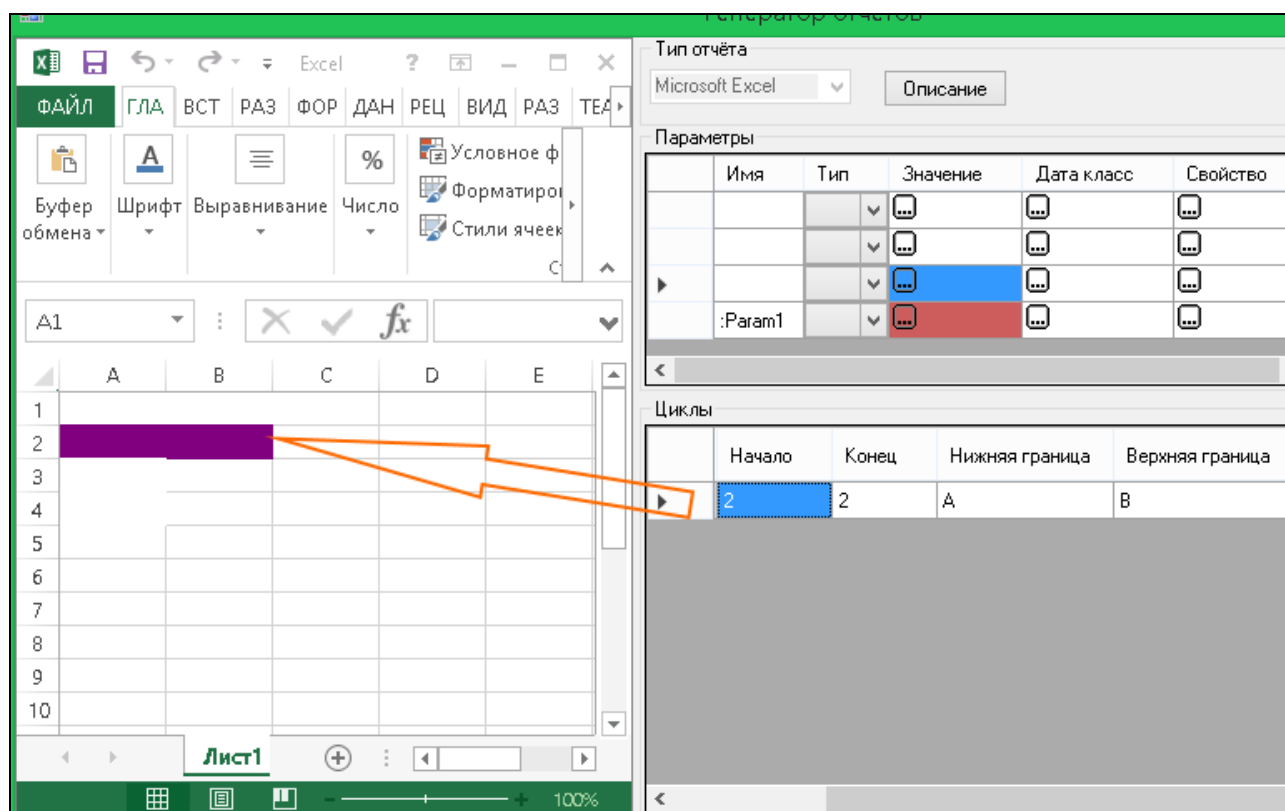
В столбцы *Нижняя граница* и *Верхняя граница* вводятся имена граничных столбцов копируемого диапазона ячеек. Эти значения допускается не указывать, в этом случае будут копироваться строки целиком.

В столбце *Сортировка* при необходимости можно указать сортировку строк цикла. Для этого нужно нажать эллипсоидную кнопку и выбрать выражения для сортировки в нужном порядке. Сортировка будет производиться по этим выражениям. Выражения для сортировки могут быть любого типа. В принципе, они даже не обязаны входить в состав цикла, т.е. отображаться в копируемом диапазоне ячеек, однако эти выражения должны работать с теми же наборами данных, что и выражения цикла. Если для какого-либо выражения требуется сортировка в обратном порядке, перед именем этого выражения добавляется знак «минус».

При заполнении параметров цикла, в качестве индикатора правильности границ цикла служит подсветка. На рисунке ниже показан пример подсветки неправильных границ цикла.

Циклы							
	Имя	Шаг	Размер	Начало	Конец	Нижняя граница	Верхняя граница
▶	cycle1	1	0	A	2	A	B

Когда цикл будет правильно заполнен, подсветка границ в строке цикла исчезнет и в excel шаблоне будет показаны границы заданного цикла.



Размер цикла

Количество строк, выводимое в цикле, называется размером цикла и отображается в поле цикла *Размер*.

Размер цикла не задается явно. Он рассчитывается при каждом запуске отчета, исходя из значений выражений, входящих в цикл. Происходит это следующим образом. Просматриваются все ячейки, входящие в область цикла, и для каждой из них анализируется выражение, на которое ссылается ячейка. Если в этом выражении есть обращения к сущностям, для каждого из таких обращений подсчитывается количество записей, возвращаемых каждой сущностью. Предположим, имеется следующее выражение:

#E1.#E2.#E3.#E4.F1(<параметры>),

где E1, E2, E3, E4 – некоторые сущности, F1 – некоторое свойство сущности E4. При заданном конкретном наборе параметров каждая сущность вернет некоторое количество записей. Обозначим эти количества как N1, N2, N3, N4 соответственно. Тогда общее количество возможных вариантов из всех сочетаний записей в разных сущностях будет выражаться следующей формулой:

$$R = \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \sum_{i_3=1}^{N_3} \sum_{i_4=1}^{N_4}$$

То есть подсчитывается количество всех возможных перестановок записей из разных сущностей.

Такое количество подсчитывается для каждого выражения, входящего в цикл. Затем выбирается максимальное количество – это и будет размер цикла. Как правило, все формулы, входящие в цикл, работают с одним и тем же набором данных и возвращают одинаковое количество, в противном случае результат, скорее всего, будет бессмысленным.

В реальности чаще всего количество записей во всех сущностях, кроме одной, равно единице. Более сложные запросы, в которых множественный отбор идет по нескольким сущностям, в реальных отчетах практически не используются из-за сложности представления и отображения информации. В этом случае размер цикла просто равен количеству записей в этой множественной сущности.

Когда составляются формулы для ячеек, входящих в цикл, обычно нет необходимости задумываться над размером цикла. Просто пишется выражение над сущностями, в состав которого входит некоторый критерий отбора, возвращающий множественный набор, и генератор отчетов сам рассчитает и отобразит все данные по каждой записи из этого множественного набора. Необходимо только следить, чтобы во всех ячейках использовался один и тот же набор данных – с одними и теми же параметрами.

Отладка циклов

Для любого цикла на панели «Циклы» в столбце *Размер*, в реальном времени рассчитывается и выводится размер цикла для текущих значений параметров. Таким образом, не запуская отчет, сразу можно увидеть, сколько строк будет выведено для данного цикла и оценить, правильно ли определены формулы в составе цикла. При изменении параметров отчета, формул или описания цикла, размер всех циклов автоматически пересчитывается.

В столбце *Шаг* показано текущее значение цикла. Это значение можно менять, и при этом в окне «Выражения» будут изменяться значения выражений, связанных с циклом. Таким образом, последовательно меняя это значение, можно просмотреть данные для каждой из строк цикла.

Надо отметить, что номера строк в данном случае условные, поскольку в реальности для цикла может быть задана сортировка, а в отладчике данные отображаются в неотсортированном виде, поэтому данные для строки с конкретным номером в отладчике и в готовом отчете могут не совпадать.

Дополнительные выражения цикла

В описании цикла в сетке «Циклы» присутствует еще один параметр, который задает дополнительные выражения цикла. Этот параметр используется для того, чтобы иметь возможность в процессе выполнения цикла вычислять дополнительные выражения, которые напрямую не входят в цикл (т.е., не отображаются в отчете для строк цикла), но от которых могут зависеть выражения цикла. В обычной ситуации при вычислении цикла и формировании его строк на каждом шаге вычисляются только те выражения на листе настройки, которые либо отображаются для строки цикла, либо указаны в качестве выражений для сортировки строк цикла. Однако, некоторые из этих выражений могут быть достаточно сложными для вычисления их в одной формуле, или же какие-то части этих выражений одни и те же для нескольких формул. В таком случае имеет смысл выделить отдельные части выражений, входящих в цикл, в самостоятельные выражения, описать

переменные для сохранения этих частей выражений и в дальнейшем в выражениях цикла использовать эти переменные. Но если просто выделить какую-то часть выражения, используемого в цикле, в отдельное выражение, то оно не будет автоматически рассчитываться для каждого шага цикла. Чтобы добиться этого, необходимо указать ссылку на это выражение в столбце «Доп. выражения» в описании цикла.

Ссылки на дополнительные выражения цикла указываются так же, как и ссылки на выражения сортировки. Т.е. нужно нажать эллипсоидную кнопку в поле дополнительных выражений и выбрать выражения, которые должны рассчитываться на каждом шаге цикла.

Таким образом, на каждом шаге цикла рассчитываются выражения, непосредственно входящие в цикл (отображаемые на листе отчета в строках цикла), выражения, указанные в параметре сортировки цикла, и выражения, входящие в список дополнительных выражений цикла.

Системные переменные, связанные с циклами

В выражениях, находящихся в ячейках цикла, допускается использование нескольких системных переменных, связанных с циклами. Эти переменные начинаются с символа «\$». Ниже перечислены все эти переменные с кратким описанием для каждой:

- **\$CYCLESTEP** – содержит значение текущего шага цикла. При выводе отчета соответствует порядковому номеру выводимой строки;
- **\$CYCLECOUNT** – содержит размер текущего цикла, т.е. при выводе отчета соответствует количеству строк в данном цикле;
- **\$CYCLENAME** – содержит имя текущего цикла;
- **\$CYCLESTEP2** – особая переменная, которая используется только при пересечении циклов и содержит текущий шаг для вертикального цикла при его пересечении с горизонтальным. Данные понятия будут подробно описаны ниже.
- **\$CYCLECOUNT2** – особая переменная, которая используется только при пересечении циклов и содержит размер вертикального цикла при его пересечении с горизонтальным. Данные понятия будут подробно описаны ниже.

Горизонтальные и вертикальные циклы

До настоящего момента, говоря о циклах, везде подразумевалось понятие горизонтального цикла. Горизонтальный цикл – это цикл, ячейки которого копируются построчно, при этом содержимое ячеек, лежащих ниже диапазона ячеек цикла, сдвигается вниз. Иногда, достаточно редко, возникает необходимость создания отчета, в котором количество столбцов непостоянно и зависит от некоторого набора данных.

В подобных случаях используются вертикальные циклы. Они отличаются от горизонтальных тем, что ячейки диапазона цикла копируются не построчно, а по столбцам, при этом содержимое ячеек, лежащих правее диапазона ячеек цикла, сдвигается вправо. Для того, чтобы задать такой цикл, необходимо в качестве начала и конца цикла задать не номера начальной и конечной строки цикла, а названия начального и конечного столбца диапазона цикла. Соответственно, в качестве нижней и верхней границы цикла нужно задавать не названия граничных столбцов, а номера граничных строк. Во всем остальном вертикальные циклы ничем не отличаются от горизонтальных.

Пересечения циклов. Специальные параметры и переменные

Итак, циклы бывают двух видов – горизонтальные, в которых данные копируются построчно, и вертикальные, в которых данные копируются по столбцам. Для каждого из этих типов циклов пересечение с циклом аналогичного типа не имеет логического смысла и

Циклы

Имя	Шаг	Размер	Начало	Конец	Нижняя граница	Верхняя граница
c1	1	8	4	4	A	F

Свойства

Имя	c1
Шаг	1
Размер	8
Тип цикла	0
Начало	4
Конец	4
Нижняя граница	1
Верхняя граница	6
Сортировка	<input type="checkbox"/>
Дополнительные выражения	<input type="checkbox"/>

Имя

Выражения отчета (формулы)

Выражения отчета содержат формулы, с помощью которых выполняются различные операции над данными, полученными из БД, необходимые для формирования отчета. Формулы отображаются на панели «Выражения», которая состоит из списка выражений и кнопок управления.

Выражения

Имя	Выражение	Значение	Переменная

Список колонок на сетке (списке) выражений.

- *Имя* – содержит имя выражения, используемое для вывода значения выражения в excel.
- *Выражение* – содержит текст формулы или скрипта.
- *Значение* – содержит значение выражения, вычисленное по заданной формуле.
- *Переменная* – локальное имя выражения, имя под которым выражение используется в других формулах.

Список кнопок управления выражениями.

- *Добавить строку*. Кнопка добавляет новое выражение в конец списка выражений текущего шаблона. Чтобы вставить выражение в середину списка нужно использовать кнопку Insert.
- *Удалить строку*. Кнопка удаляет текущее выражение текущего шаблона.
- *Переместить вверх, Переместить вниз*. Кнопки предназначены для управления порядком следования выражений в шаблоне.

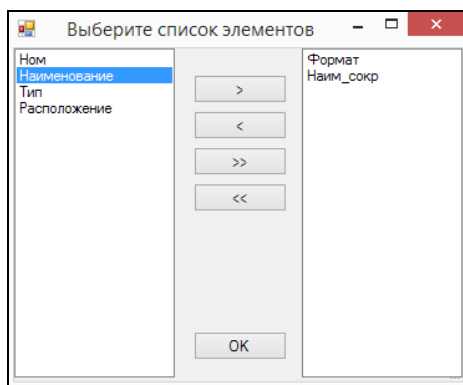
Имена выражений. Ссылки на выражения

Имена выражений используются для ссылок на выражение из шаблона excel или из циклов отчета. Имена должны быть уникальными в пределах шаблона отчета, могут содержать любые русские и латинский буквы, цифры и знак подчеркивания (пробелы и иные символы не допустимы). Имена необходимы в нескольких случаях:

- Если значение выражения является результатом, т.е. должно быть напечатано в отчете.
- Если выражение участвует в сортировке цикла. Только выражение, имеющее *Имя*, может быть выбрано для участия в сортировке.
- Если выражение является доп. выражением цикла. Только выражение, имеющее *Имя*, может быть использовано как дополнительное выражение цикла.

Все остальные выражения, которые не выводятся на печать, не являются дополнительными выражениями цикла и не участвуют в сортировке цикла, могут не иметь *Имени*. При желании можно задать *Имя* и этим выражениям тоже, но необходимости в этом нет.

Установка ссылки на выражение из цикла для сортировки цикла по этому выражению выполняется на панели циклов. Нужно нажать эллипсоидную кнопку в поле *Сортировка* в строке нужного цикла. Откроется окно, состоящее из двух частей, разделенных кнопками >, <, >>, <<, ОК.

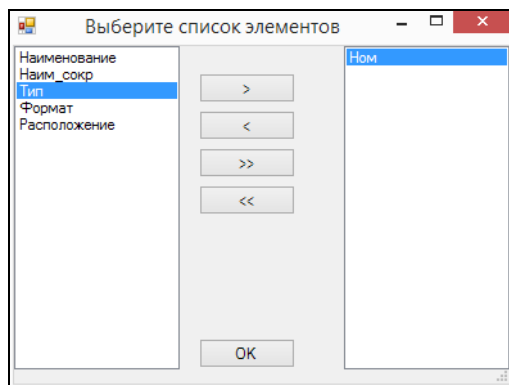


В левой части окна показан список *Имен* выражений, которые могут быть выбраны для сортировки цикла. Выражения, для которых *Имя* не задано, в этом списке не отображаются (их нельзя выбрать для сортировки цикла). В правой части окна показан список *Имен* выражений, которые участвуют в сортировке цикла (уже выбраны).

Чтобы включить выражение в сортировку, нужно в левой части окна встать курсором на его *Имя* и нажать кнопку >. Выражение переместится в правую часть окна (в список выбранных). Чтобы удалить выражение из сортировки, нужно в правой части окна встать курсором на его *Имя* и нажать кнопку <. Выражение переместится в левую часть окна (в список не выбранных).

Кнопка >> служит для добавления всех выражений в сортировку цикла. Кнопка << необходима для очистки сортировки. По кнопке ОК сохраняются все изменения сортировки.

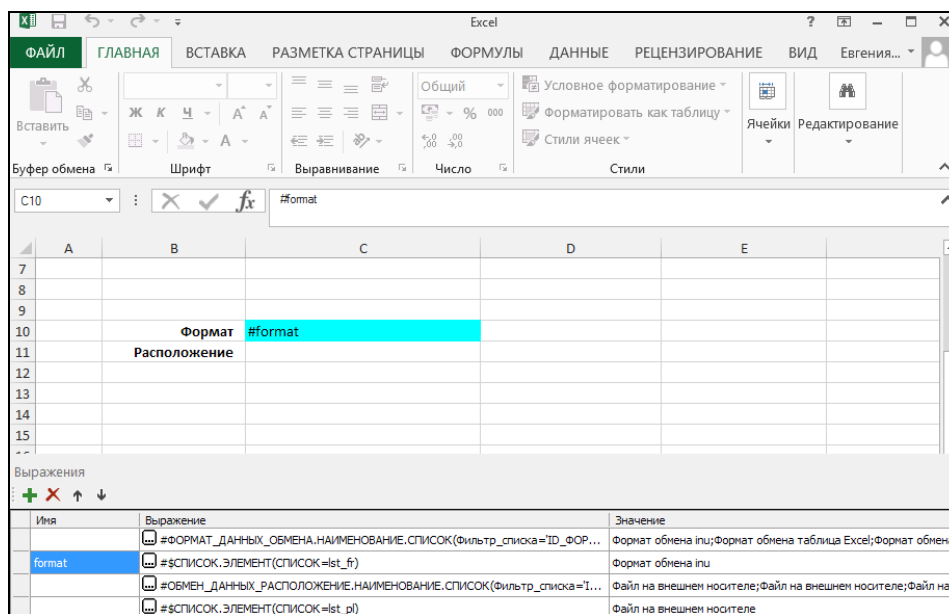
Установка ссылки на выражение из цикла для расчета этого выражения на каждом шаге цикла выполняется также на панели циклов. Нужно нажать эллипсоидную кнопку в поле *Доп.выражение* в строке нужного цикла. Откроется окно, аналогичное окну выбора выражений для сортировки.



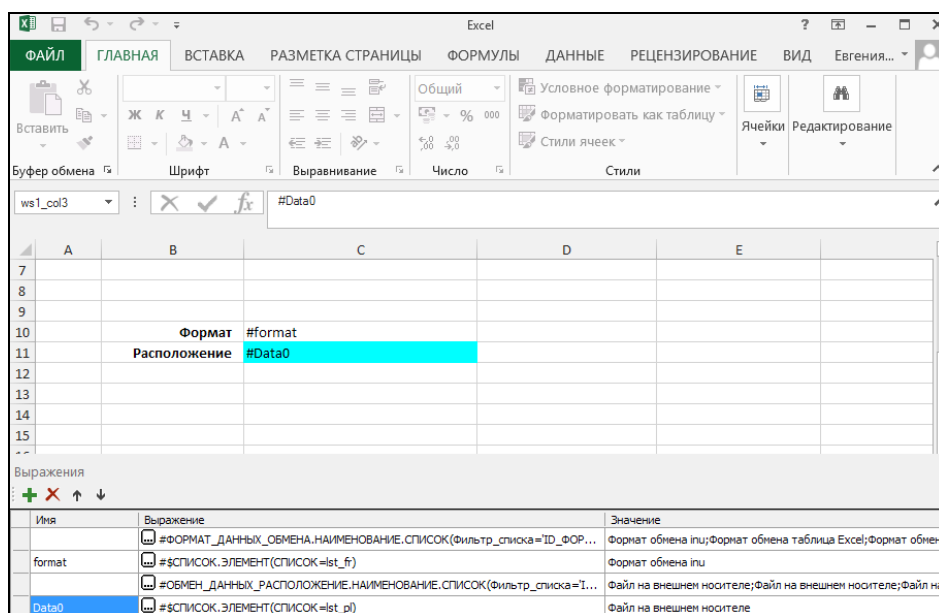
В левой части окна показан список *Имен* выражений, которые могут быть включены в список дополнительных выражений цикла. Выражения, для которых *Имя* не задано, в этом списке не отображаются (их нельзя выбрать в качестве дополнительных выражений цикла). В правой части окна показан список *Имен* выражений, которые уже являются дополнительными выражениями цикла.

Управлять набором доп. выражений цикла можно с помощью кнопок >, <, >>, <<, ОК.

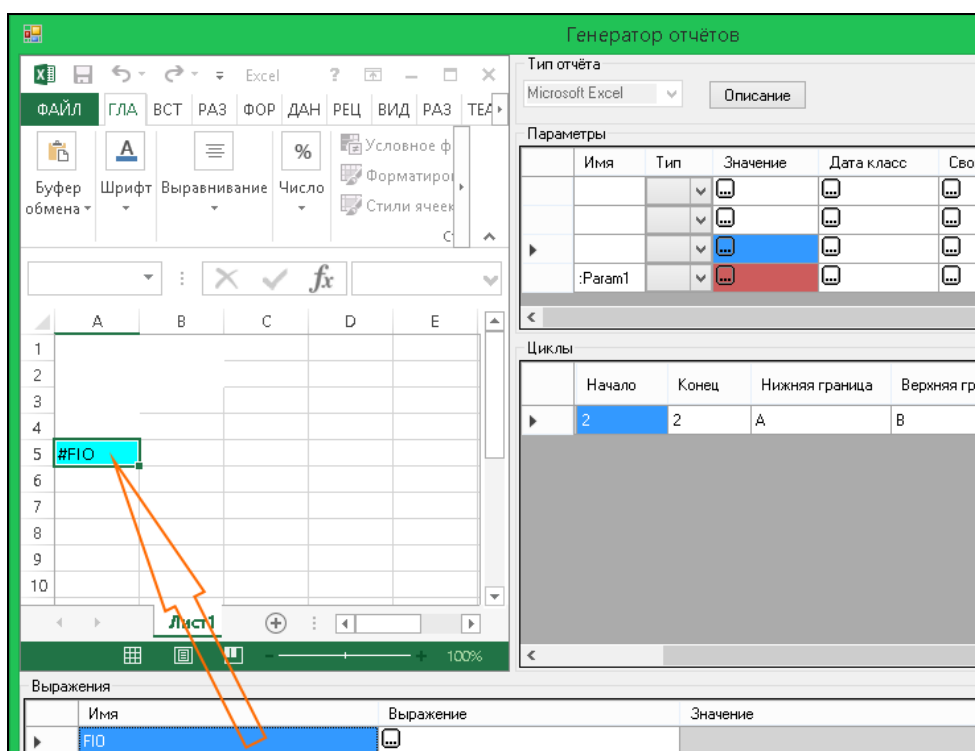
Установка ссылки на выражение из ячейки Excel для вывода значения выражения в отчет выполняется с помощью механизма перетаскивания. Для установки ссылки между выражением и ячейкой Excel нужно кликнуть мышью *Имя* нужного выражения, и, не отпуская клавишу мыши, «перетащить» его в нужную ячейку листа Excel. После того, как клавиша мыши будет отпущена, в заданную ячейку Excel будет автоматически помещена ссылка на выражение, в которую при выводе отчета будет записано рассчитанное значение. При этом, ссылка будет видна в ячейке Excel в виде #*Имя* выражения.



Если выражение, для которого устанавливается ссылка не имеет имени, оно будет присвоено автоматически: *Data0*, *Data1*, ...



Список выражений, как и список циклов, взаимодействует с панелью excel. При движении по списку выражений, в случае, если выражение выводится в отчет, будет подсвечена ячейка (ячейки), в которых это выражение встречается.



Переменные выражений


Для того, чтобы использовать значение любого выражения в дальнейших вычислениях, его нужно сохранить в какую-либо переменную. Для этого необходимо на панели выражений ввести имя переменной в поле *Переменная*. В дальнейшем имя этой переменной можно будет использовать в любом выражении. Если выражение не планируется использовать в других формулах, то переменную для этого выражения можно не задавать.

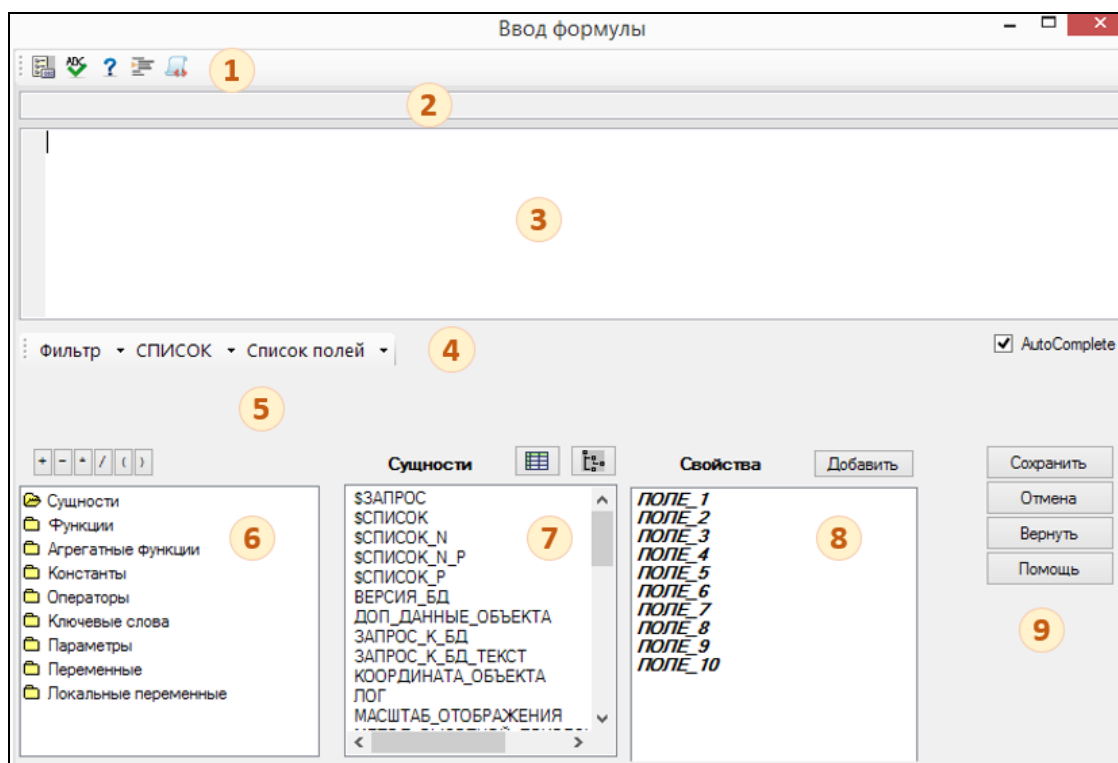
Выражения и их значения

Выражение – это сама формула или скрипт (группа формул), написанная на языке построения отчетов *S_Технологии*. Выражения вводятся в специальной форме ввода формул, подробнее об этой форме, формулах и языке построения отчетов см. ниже.

Значение выражения рассчитывается автоматически при каждом запуске отчета, исходя из формул *Выражения* и значений параметров. Для любого выражения в реальном времени рассчитывается и выводится его *Значение* для текущих формул и значений параметров. Таким образом, не запуская отчет, сразу можно увидеть результат выражения и оценить правильно ли определены формулы выражений. При изменении параметров отчета или формул *Значения* выражений автоматически пересчитываются.

Форма ввода формул

Форма «Ввод формулы» служит для конструирования *Выражений* – формул и групп формул (скриптов) – с использованием сущностей БД и встроенных функций генератора отчетов. Форма вызывается с панели выражений по клику на эллипсоидную кнопку  в колонке «Выражение». Вид формы представлен на рисунке ниже.

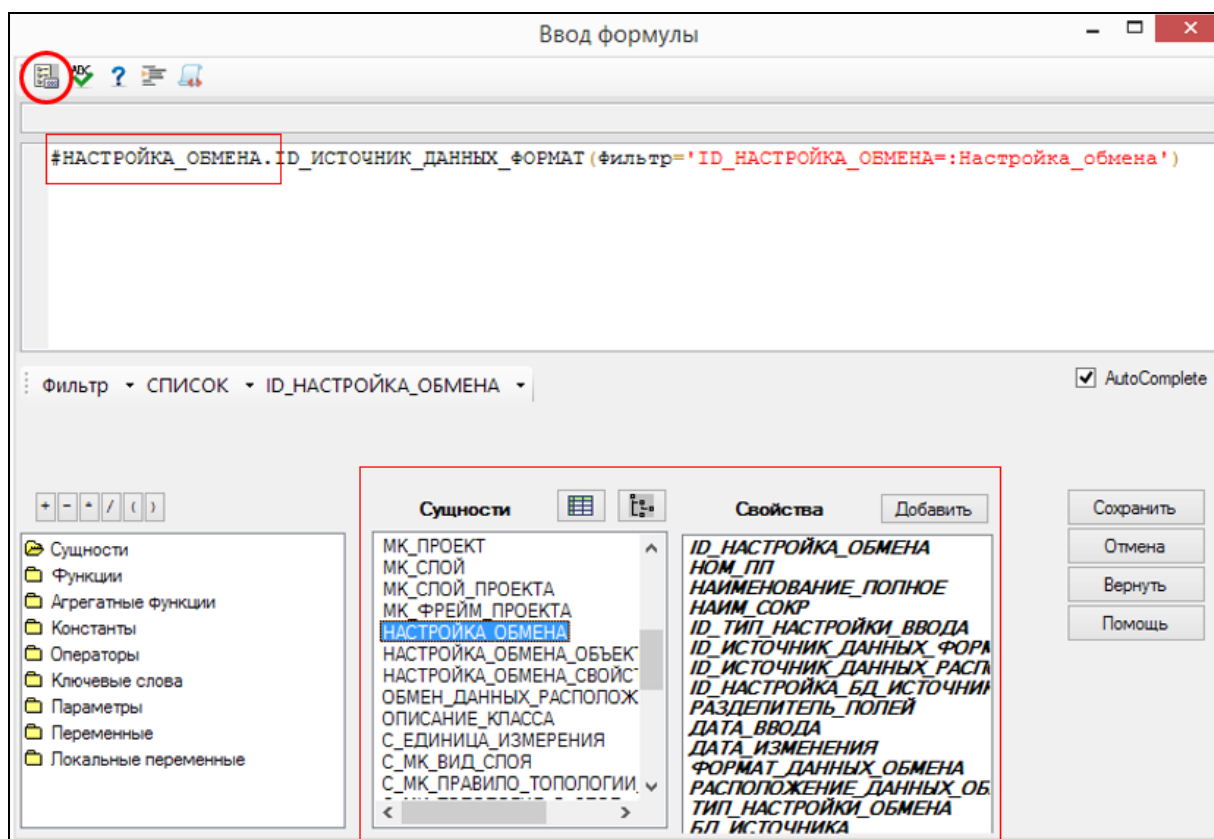


На форме можно выделить следующие функциональные области:

1. Панель с кнопками справочного назначения и форматирования формул.



- *Найти сущность для текущей формулы*. Нажатие кнопки устанавливает курсор на сущность, используемую в формуле.



- Проверить/получить результат. Нажатие кнопки отображает результат (или ошибку) на панели результата.



- Справка по выделенной функции. Нажатие кнопки выводит на панель справки справочную информацию по функции, выделенной мышью в редакторе ввода формул.

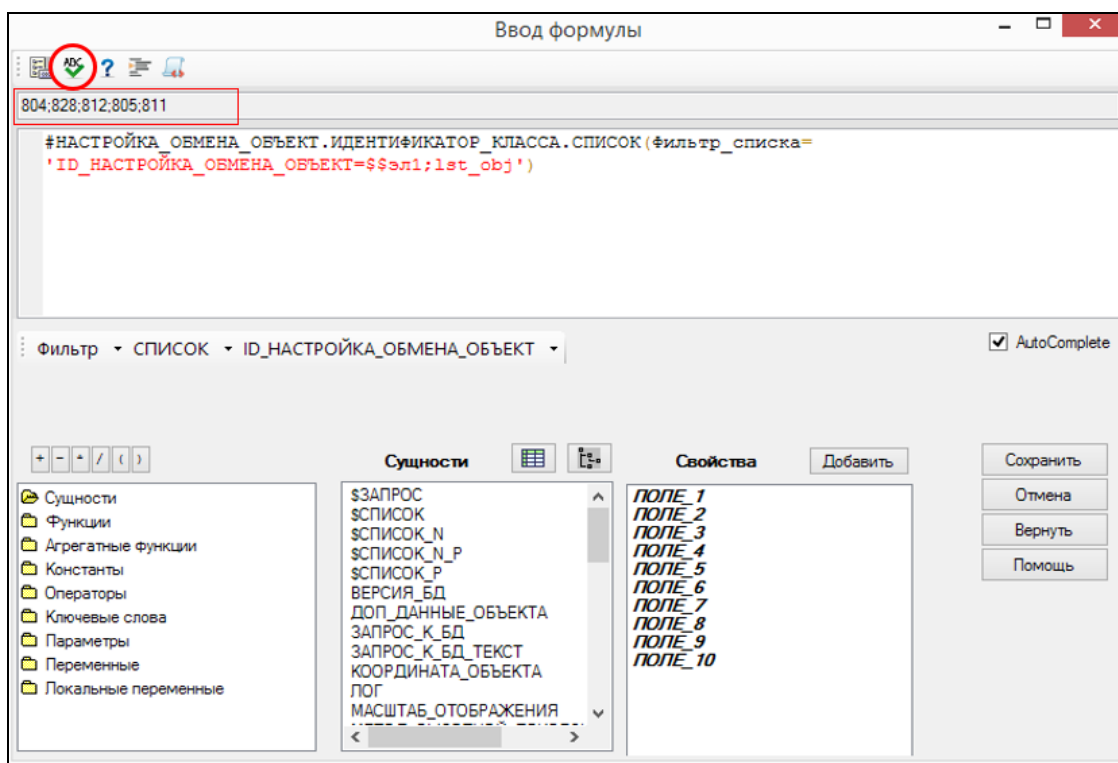


- Автоформат скрипта. Нажатие кнопки форматирует скрипт, написанный в окне редактора ввода формулы.

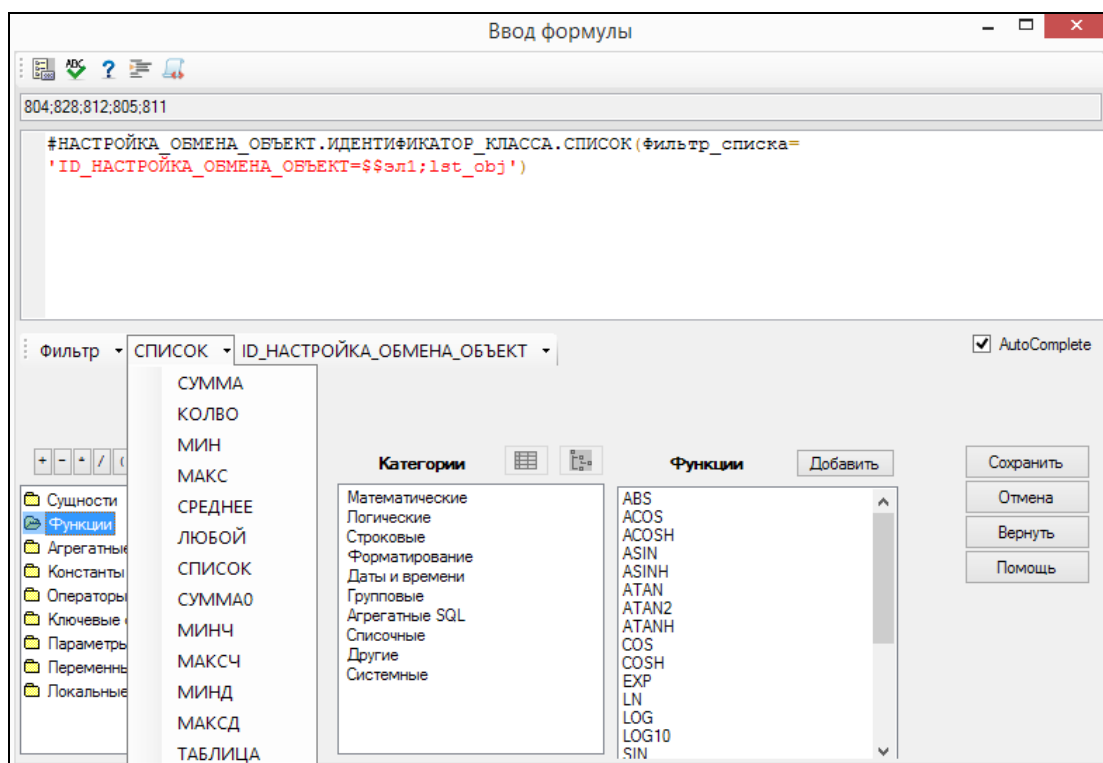


- Показать сниппеты. Нажатие кнопки отображает панель сниппетов для выбора и использования в формуле (скрипте).

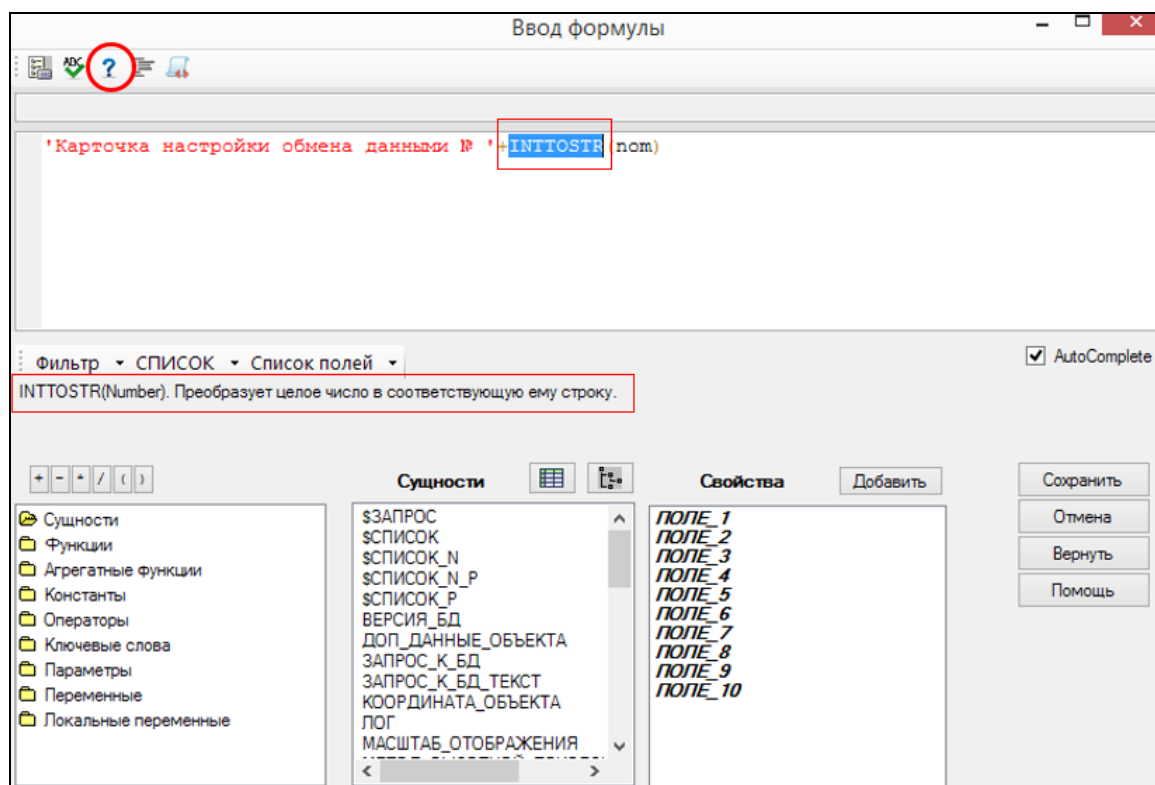
2. **Панель результата.** Позволяет оперативно увидеть результат текущей формулы (скрипта), не выходя из формы ввода формул.



3. **Редактор ввода формулы.** Служит для отображения и редактирования текущего выражения шаблона (формулы, скрипта).
4. **Панель-помощник для простых формул.** Функционал панели упрощает ввод простых формул. Здесь можно быстро выбрать параметр или агрегатную функцию из числа часто используемых (подробнее о параметрах и функциях см. ниже), можно выбрать любое поле текущей сущности одним кликом выши, здесь можно включить или выключить функцию AutoComplete.



5. **Панель справки.** Отображает текст подсказки для выделенной функции.



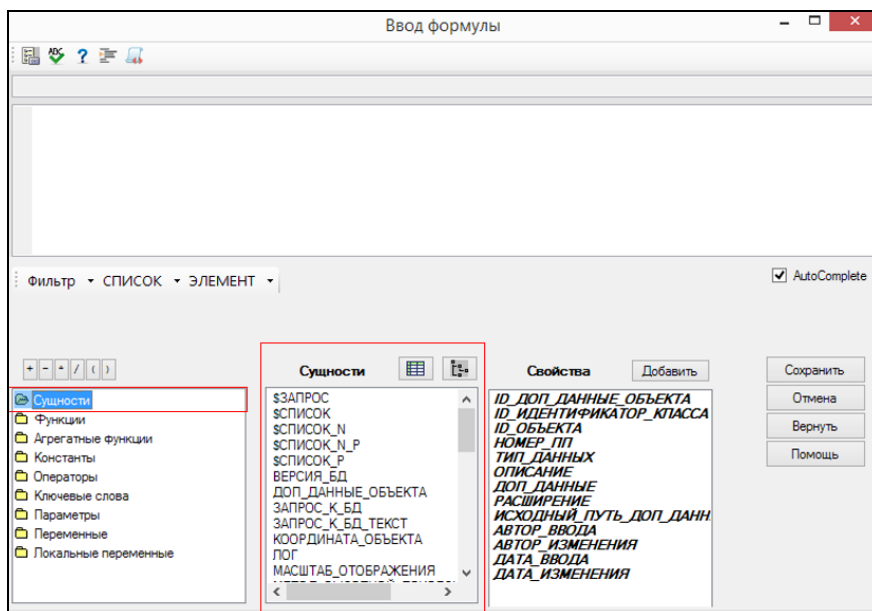
6. **Панель со списком объектов формул.** Отображает объекты, доступные для использования в выражениях: сущности, функции, константы, параметры, переменные и т.д.

7. **Панель со списком элементов объектов.** Отображает элементы текущего объекта: список сущностей, список категорий функций, список параметров и т.д.
8. **Панель со списком свойств элементов объектов.** Отображает свойства текущего элемента текущего объекта: список полей текущей сущности, список функций текущей категории и т.д.
9. **Кнопки управления формой.**
 - *Сохранить.* Закрытие формы с сохранением изменений.
 - *Отмена.* Закрытие формы без сохранения изменений.
 - *Вернуть.* Отмена последнего действия.
 - *Помощь.* Вызов справки.

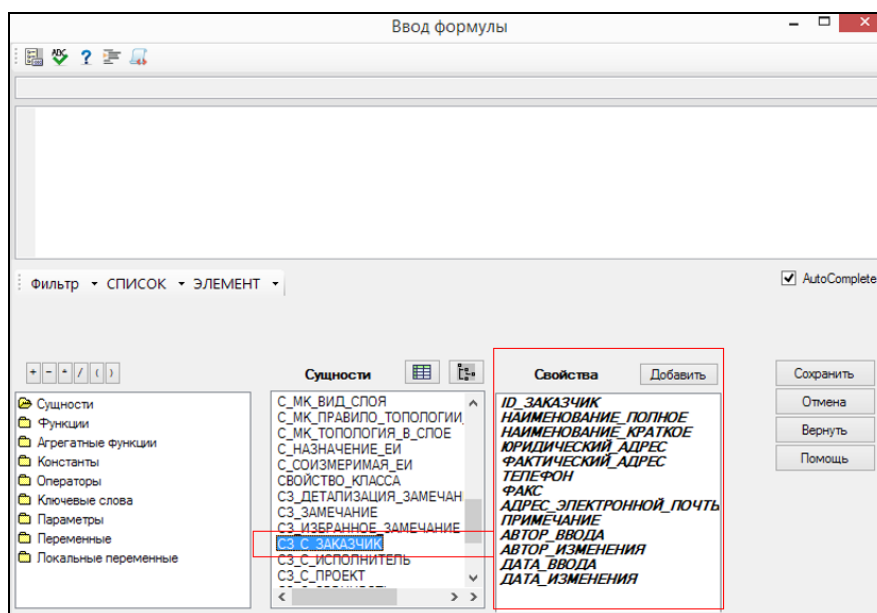
Определение сущности

Основным понятием, используемым в языке построения отчетов, является *Сущность*. *Сущность* – это некоторый объект в используемой базе данных, относящийся к рассматриваемой предметной области. Каждая сущность имеет свое имя, по смыслу совпадающее или близкое к тому объекту, который она описывает. Например, сущность для замечаний называется «СЗ_ЗАМЕЧАНИЕ», сущность для единиц измерения называется «С_ЕДИНИЦА_ИЗМЕРЕНИЯ» и т.д. Имена сущностей для генератора отчетов определяют разработчик приложения, задавая FormulaName для таблиц БД в модулях сущностей.

Список всех доступных сущностей можно увидеть в форме ввода формул, выбрав на панели объектов формул строку *Сущности* (она активна по умолчанию). Тогда на панели элементов объектов формул появится список всех сущностей проекта, отсортированных по алфавиту.



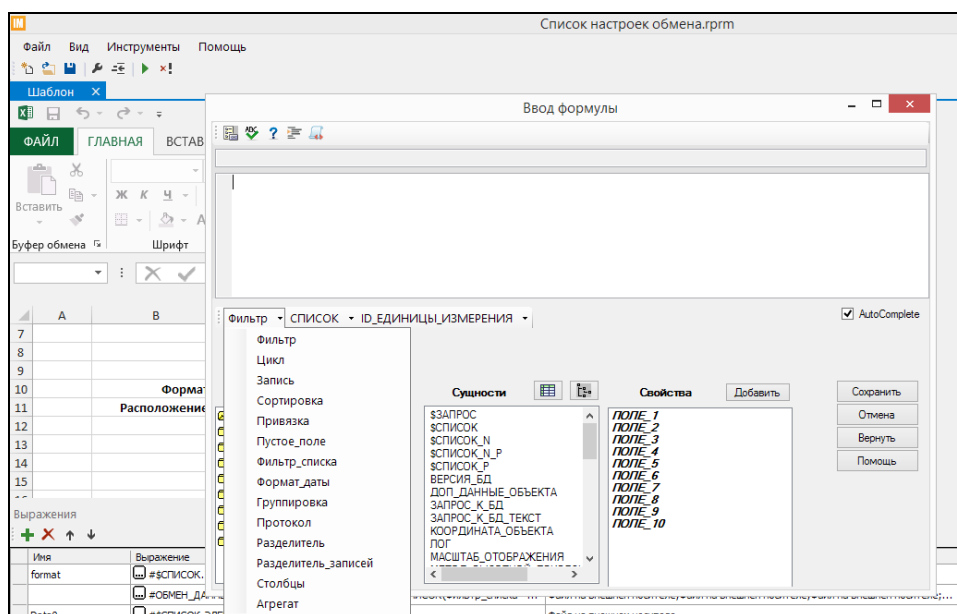
Каждая сущность имеет свой фиксированный набор свойств. Имена свойств для генератора отчетов также определяют разработчик приложения, задавая FormulaName для полей таблиц БД в модулях сущностей. Список свойств сущности можно увидеть в форме ввода формул, выбрав в списке сущностей нужную сущность. Тогда на панели свойств элементов объектов формул появится список всех свойств текущей сущности.



Основным элементом выражения языка отчетов является обращение к некоторому свойству некоторой сущности. В простом варианте это обращение выглядит следующим образом:

#<Сущность>.<Свойство>(⟨параметры отбора⟩)

С помощью параметров отбора можно задавать практически любые критерии отбора. Список доступных параметров отбора можно увидеть на панели-помощнике, раскрыв первое меню панели, обозначенное как *Фильтр*.



Первый параметр этого списка *Фильтр* наиболее часто применяется для отбора и, как правило, без него не обходится ни один отчет. Значением этого параметра является строка с выражением отбора. Выражение отбора по форме аналогично тому, который используется для фильтров в Excel, только в качестве источника данных выступают не ячейки, а свойства сущности. Вот один из примеров такого отбора:

```
#С_ЕДИНИЦА_ИЗМЕРЕНИЯ.КОД_ЕДИЗМ(Фильтр='НАИМЕНОВАНИЕ="Метр"')
```

Данное выражение выбирает код единицы измерения с наименованием «Метр». Если таких единиц измерения окажется несколько, будет взят какой-то один из их числа.

Еще один пример, немного более сложный:

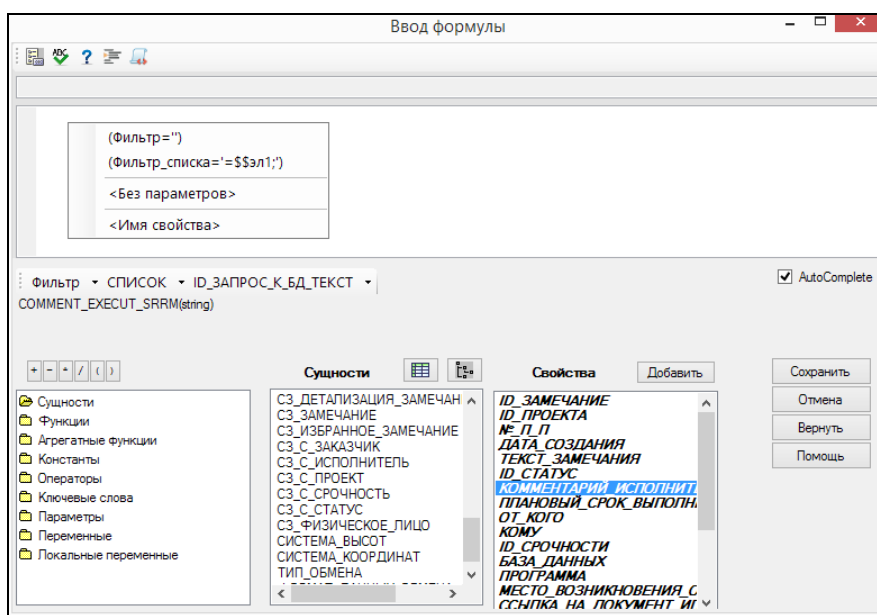
```
#СЗ_ФИЗИЧЕСКОЕ_ЛИЦО.АДРЕС_ЭЛЕКТРОННОЙ_ПОЧТЫ(Фильтр='ДАТА_РОЖДЕНИЯ>"01.01.1990" and ОБРАЗОВАНИЕ="высшее" and ISNULL(КОНТАКТНЫЙ_ТЕЛЕФОН)')
```

Данное выражение отбирает адрес электронной почты физического лица, родившегося не ранее 1990 г. и имеющего высшее образование, для которого не указан контактный телефон.

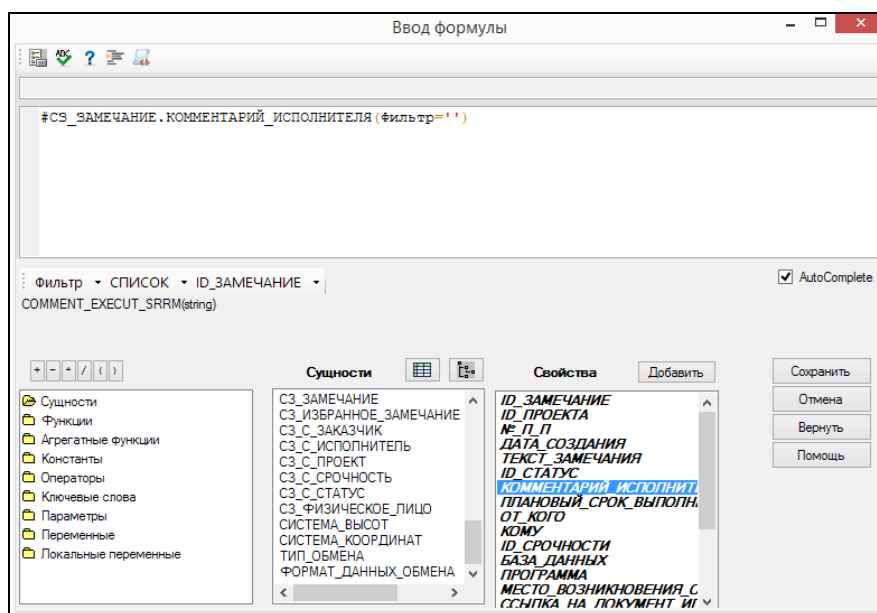
Для выражения фильтра допустимы сколь угодно сложные выражения над свойствами текущей сущности, включая все встроенные функции (описаны ниже).

Ввод формулы

Ввод формул автоматизирован. Имена сущностей и их свойств не нужно набирать вручную. Для того чтобы использовать свойство сущности в формуле, нужно установить курсор на это свойство, и перетащить мышью в верхнюю часть окна на панель редактора ввода формулы. После того, как кнопка мыши будет отпущена, появится меню, из которого нужно выбрать вариант параметров отбора.



Выбрав параметр, получим шаблон выражения, который уже содержит имя сущности, свойство сущности и, возможно, параметр отбора. Нужно добавить только условия для параметра отбора.

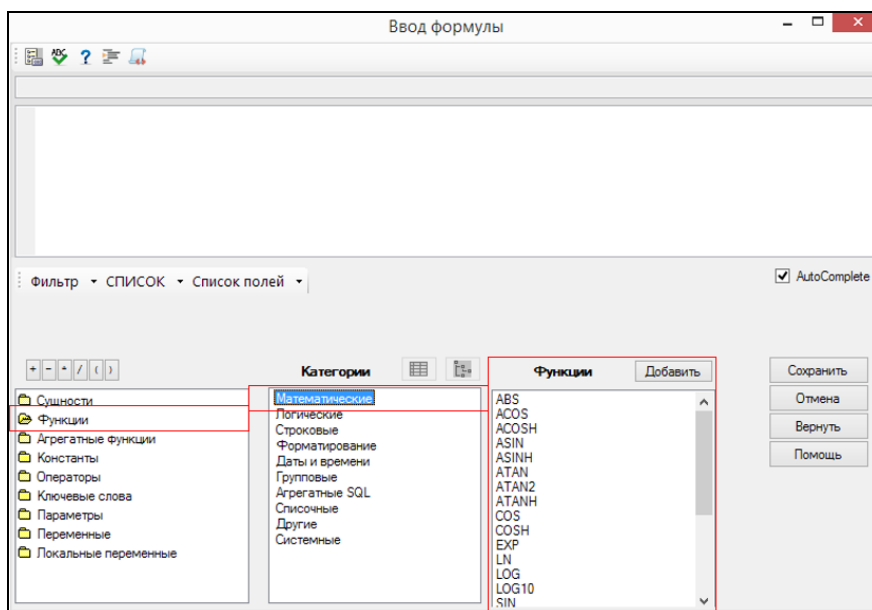


В условиях параметров отбора можно использовать следующие конструкции.

- *Свойства сущности.* Имена свойств можно набирать вручную, добавлять двойным кликом по нужному свойству на списке свойств или перетаскивать в нужное место формулы из списка свойств сущности, выбрав из появившегося меню пункт «<Имя свойства>». Ниже приведен пример условия отбора с использованием свойства ID_ПРОЕКТА.

#СЗ_ЗАМЕЧАНИЕ.КОММЕНТАРИЙ_ИСПОЛНИТЕЛЯ(Фильтр='ID_ПРОЕКТА=5')

- *Функции языка построения отчетов.* Список всех доступных функций можно увидеть, выбрав на панели объектов формул строку *Функции*. Тогда на панели элементов объектов формул появится список категорий функций проекта. Каждая категория имеет свой набор функций. Набор функций определен разработчиком S_Технологии. Список функций каждой категории можно увидеть, выбрав в списке категорий нужную категорию. Тогда на панели свойств элементов объектов формул появится список всех функций текущей категории.



Имена функций можно набирать вручную, а также добавлять двойным кликом по функции или перетаскивать в нужное место формулы из списка функций. Ниже приведен пример условия отбора с использованием строковой функции LENGTH.

#СЗ_ЗАМЕЧАНИЕ.КОММЕНТАРИЙ_ИСПОЛНИТЕЛЯ(Фильтр='LENGTH(ТЕКСТ_ЗАМЕЧАНИЯ)>0')

Подробнее о функциях и их использовании см. ниже, в главе [Встроенные функции](#).

- *Агрегатные функции языка построения отчетов.* Список всех доступных агрегатных функций можно увидеть, выбрав на панели объектов формул строку *Агрегатные функции*. Тогда на панели элементов объектов формул появится список категорий агрегатных функций проекта. Каждая категория имеет свой набор агрегатных функций. Набор агрегатных функций определен разработчиком S_Технологии. Список агрегатных функций каждой категории можно увидеть, выбрав в списке категорий нужную категорию. Тогда на панели свойств элементов объектов формул появится список всех агрегатных функций текущей категории.

Имена агрегатных функций можно набирать вручную, а также добавлять двойным кликом по функции, перетаскивать в нужное место формулы из списка агрегатных функций или выбирать на панели-помощнике, раскрыв второе меню панели, обозначенное как *СПИСОК*. Ниже приведен пример условия отбора с использованием строковой функции КОЛВО.

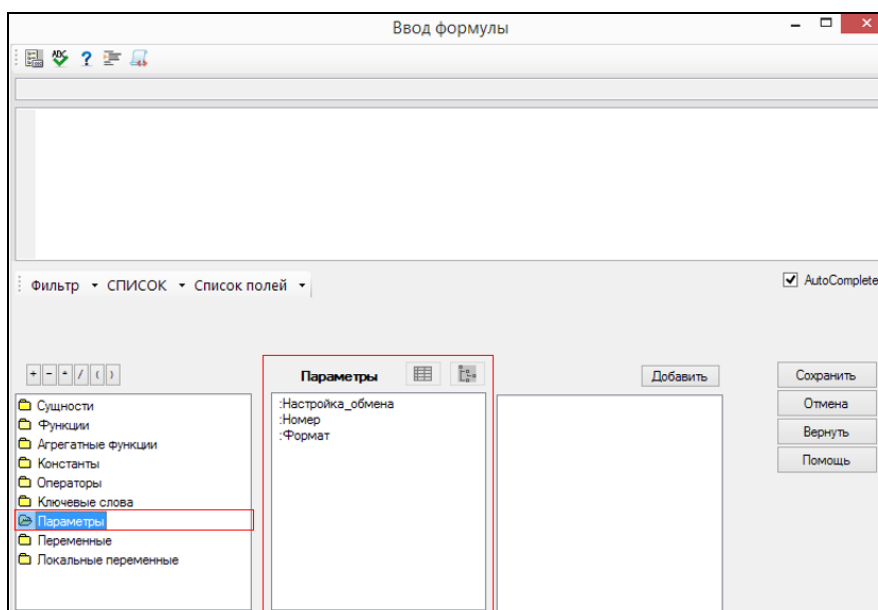
#СЗ_ЗАМЕЧАНИЕ.КОММЕНТАРИЙ_ИСПОЛНИТЕЛЯ.КОЛВО(Фильтр='№_П_П=10')

Подробнее об агрегатных функциях и их использовании см. ниже, в главе [Агрегатные функции](#).

- *Константы, Операторы, Ключевые слова.* Список всех доступных констант, операторов и ключевых слов можно увидеть, выбирая на панели объектов формул строки *Константы, Операторы, Ключевые слова* соответственно. Тогда на панели элементов объектов формул появится список типов констант, категорий операторов или ключевых слов. Каждый тип имеет свой набор констант, а каждая категория – свой набор операторов. Набор констант, операторов и ключевых слов определен разработчиком S_Технологии. Список констант или операторов каждой категории можно увидеть, выбрав в списке типов/категорий нужный тип/катеорию. Тогда на панели свойств элементов объектов формул появится список всех констант или операторов текущего типа/категории.

Имена констант, операторов и ключевых слов можно набирать вручную, а также добавлять двойным кликом или перетаскивать в нужное место формулы из списка констант, операторов или ключевых слов.

- *Параметры отчета.* Можно добавлять в выражения параметры отчета. Список параметров отчета можно увидеть на [панели параметров](#) или в форме ввода формул, выбрав на панели объектов формул строку *Параметры*. Тогда на панели элементов объектов формул появится список параметров текущего отчета. Набор параметров определяется автором шаблона отчета (разработчиком или конечным пользователем проекта) для каждого шаблона отчета отдельно.

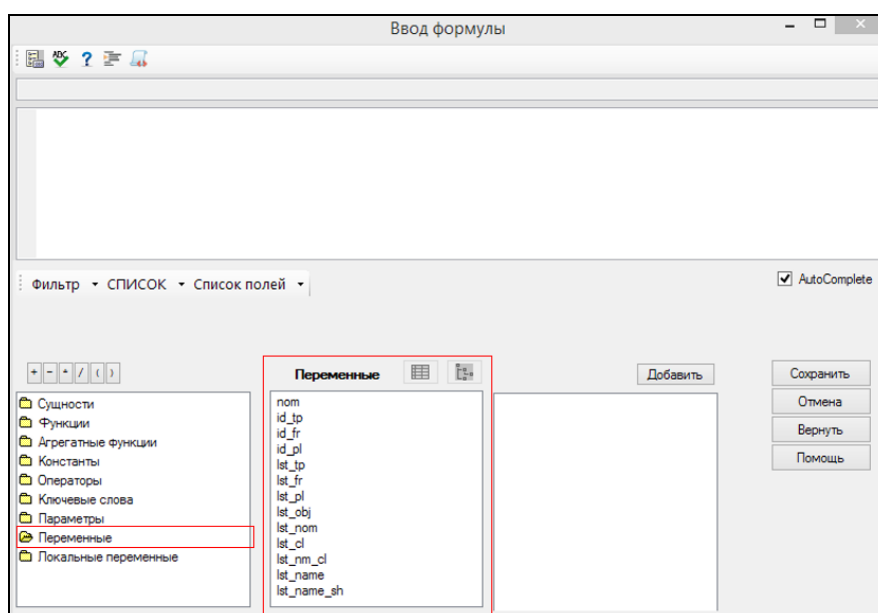


Имена параметров можно набирать вручную, а также добавлять двойным кликом по параметру или перетаскивать в нужное место формулы из списка параметров. Ниже приведен пример условия отбора с использованием параметра :Настройка_обмена.

```
#НАСТРОЙКА_ОБМЕНА.НОМ_ПП(Фильтр='ID_НАСТРОЙКА_ОБМЕНА=:Настройка_обмена')
```

Подробнее о параметрах и их использовании см. выше, в главе [Параметры отчета](#).

- **Переменные отчета.** Можно добавлять в выражения имена переменных, определенных внутри отчета. Список переменных отчета можно увидеть на [панели выражений](#) в колонке *Переменная* или в форме ввода формул, выбрав на панели объектов формул строку *Переменные*. Тогда на панели элементов объектов формул появится список переменных текущего отчета. Набор переменных определяется автором шаблона отчета (разработчиком или конечным пользователем проекта) для каждого шаблона отчета отдельно.



Имена переменных можно набирать вручную, а также добавлять двойным кликом по переменной или перетаскивать в нужное место формулы из списка переменных. Ниже приведен пример условия отбора с использованием переменной id_tp.

```
#ТИП_ОБМЕНА.НАИМЕНОВАНИЕ(Фильтр='ID_ТИП_ОБМЕНА=id_tp')
```

Подробнее о переменных и их использовании см. выше, в главе [Переменные выражений](#).

- *Локальные переменные.* Список локальных переменных отчета можно увидеть, выбрав на панели объектов формул строку *Локальные переменные*. Тогда на панели элементов объектов формул появится список переменных текущего отчета. Набор локальных переменных включает [системные переменные](#), набор которых определяет разработчик, а также локальные переменные, объявленные в скриптах текущего шаблона отчета автором шаблона.

Имена локальных переменных можно набирать вручную, а также добавлять двойным кликом по переменной или перетаскивать в нужное место формулы из списка локальных переменных. Ниже приведен пример условия отбора с использованием локальной переменной \$CYCLESTEP.

```
#СЗ_С_ИСПОЛНИТЕЛЬ.ФИО(Фильтр='№_П_П=$CYCLESTEP')
```

Основные элементы языка

Агрегатные функции

Если при обращении к свойству сущности параметры отбора заданы так, что возвращают более чем одну запись, выбирается произвольная запись. В большинстве случаев такой отбор не имеет смысла. Однако существует одно важное исключение. Иногда бывает необходимо посчитать какие-либо суммарные данные по группе записей, отобранных из сущности. Например, найти количество невыполненных заданий за период или выбрать максимальный номер настройки экспорта.

Для осуществления подобного рода вычислений используются агрегатные функции. Эти функции обрабатывают каждое значение свойства для каждой отобранной записи из сущности, и возвращают некий суммарный результат. Выражение для агрегатной функции записывается так же, как и обычное обращение к свойству сущности, но дополнительно после имени свойства через символ «.» добавляется имя агрегатной функции.

```
#СЗ_ЗАМЕЧАНИЕ.ID_ЗАМЕЧАНИЕ.КОЛВО(Фильтр='DATEISEMPTY(ДАТА_ВЫПОЛНЕНИЯ)')
```

```
#НАСТРОЙКА_ОБМЕНА.НОМ_ПП.МАКС(Фильтр='ID_ТИП_НАСТРОЙКИ_ВВОДА=3')
```

Вызов агрегатной функции осуществляется указанием ее после имени свойства сущности через символ «.». Агрегатная функция обрабатывает каждое значение свойства для каждой отобранной записи из сущности и возвращает суммарный результат. В настоящий момент определено 13 агрегатных функций.

- **СПИСОК** – возвращает символьную строку, в которой перечислены через точку с запятой все значения указанного свойства для каждой записи. В дальнейшем этот список может быть обработан с помощью специальных встроенных функций для работы со списками, которые будут рассмотрены [ниже](#).

- **СУММА** – возвращает сумму значений по указанному свойству. Эта функция применима только к данным числового типа.
- **СУММА0** – возвращает сумму значений по указанному свойству. Эта функция применима только к данным числового типа. В случае отсутствия данных возвращает 0 вместо NULL.
- **КОЛВО** – возвращает количество неповторяющихся значений указанного свойства. Необходимо обратить особое внимание, что этот результат может не совпадать с количеством отобранных записей из сущности.
- **МИН** – возвращает минимальное значение указанного свойства. Эта функция применима к данным любого типа, при этом строки сравниваются в алфавитном порядке, дата и время – в календарном порядке, для логических значений ложь (FALSE) считается меньше, чем истина (TRUE).
- **МАКС** – возвращает максимальное значение указанного свойства по тем же правилам, что и функция МИН.
- **СРЕДНЕЕ** – возвращает среднее значение указанного свойства. Эта функция применима к данным любого типа, кроме символьных, при этом для числовых данных, даты и времени считается среднее арифметическое, для логических данных возвращается истина (TRUE), если количество истинных значений (TRUE) превышает количество ложных (FALSE).
- **ЛЮБОЙ** – возвращает произвольное значение указанного свойства. Используется в некоторых специальных случаях, которые здесь рассматриваться не будут.
- **МИНЧ** – возвращает минимальное значение указанного свойства, рассмотренного в качестве целого числа.
- **МАКСЧ** – возвращает максимальное значение указанного свойства, рассмотренного в качестве целого числа.
- **МИНД** – возвращает минимальное значение указанного свойства, рассмотренного в качестве даты. Строки сравниваются в календарном порядке.
- **МАКСД** – возвращает максимальное значение указанного свойства, рассмотренного в качестве даты. Строки сравниваются в календарном порядке.
- **ТАБЛИЦА** – возвращает символьную строку, в которой перечислены через точку с запятой значения всех свойств для каждой записи, записи разделены символом #.

Встроенные функции

Объект формул «Функции» позволяет вывести список встроенных функций, которые могут использоваться в выражениях. Эти функции допускается использовать в любом месте выражения, в том числе и в выражении для параметра отбора «Фильтр». Все функции разбиты на несколько категорий. Ниже приведено краткое описание этих категорий и входящих в них функций.

Математические функции

Категория «Математические» объединяет стандартные математические функции. Для тригонометрических функций, использующих или возвращающих значение угла, этот угол задается в радианах (долях числа π).

- **ABS(x)** – возвращает модуль числа x;
- **ACOS(x)** – возвращает арккосинус числа x;
- **ACOSH(x)** – возвращает гиперболический арккосинус числа x;
- **ASIN(x)** – возвращает арксинус числа x;
- **ASINH(x)** – возвращает гиперболический арксинус числа x;

- **ATAN(x)** – возвращает арктангенс числа x ;
- **ATAN2(y, x)** – возвращает арктангенс отношения y/x в виде числа от $-\pi$ до π . Знак результата задает квадрант, в котором расположен данный угол;
- **ATANH(x)** – возвращает гиперболический арктангенс числа x ;
- **COS(x)** – возвращает косинус числа x ;
- **COSH(x)** – возвращает гиперболический косинус числа x ;
- **EXP(x)** – возвращает значение основания натурального логарифма (число e), возведенное в степень x ;
- **LN(x)** – возвращает натуральный логарифм числа x ;
- **LOG(x, base)** – возвращает логарифм числа x по основанию $base$;
- **LOG10(x)** – возвращает десятичный логарифм числа x ;
- **SIN(x)** – возвращает синус числа x ;
- **SINH(x)** – возвращает гиперболический синус числа x ;
- **TAN(x)** – возвращает тангенс числа x ;
- **TANH(x)** – возвращает гиперболический тангенс числа x ;
- **DEGREE(x)** – переводит угол x из радиан в градусы;
- **SIGN(x)** – возвращает знак числа x : -1 для отрицательных чисел, 1 для положительных чисел, 0 для нуля;
- **SQRT(x)** – возвращает квадратный корень числа x ;
- **ROUND(x, [n])** – округляет значение x до ближайшего целого, если аргумент n не задан или равен нулю. Если $n > 0$, округляет число до n знаков после запятой (т.е., например, $ROUND(1453.64591, 3) = 1453.646$. Если $n < 0$, округляет число таким образом, что первые $-n$ знаков перед запятой становятся нулями (т.е., например, $ROUND(1453.64591, -2) = 1500$;
- **TRUNC(x)** – отбрасывает от вещественного числа x дробную часть, возвращая целое число;
- **PI()** – возвращает отношение длины окружности к ее диаметру (число π). Аргумент функции может быть произвольным;
- **RAD(x)** – переводит угол x из градусов в радианы;
- **FACTORIAL(N)** – возвращает значение факториала для целого числа N .

Логические функции

Категория «Логические» объединяет функции, результат которых зависит от некоторого логического выражения. В нее входят следующие функции.

- **IF(Condition, Value1, Value2)** – проверяет значение логического выражения $Condition$, и если оно равно истине, возвращает в качестве результата значение $Value1$, в противном случае возвращает значение $Value2$. Типы значений для $Value1$ и $Value2$ могут быть любыми, но обязаны совпадать у обоих значений;
- **DATEISEMPTY(Date)** – возвращает истину, если значение $Date$ представляет собой пустую (нулевую) дату;
- **IFNULL(Value1, Value2)** – проверяет значение $Value1$ на совпадение с пустым значением ($null$), и в случае истины, возвращает в качестве значения результат значения $Value2$. Иначе в качестве результата возвращает само значение $Value1$. Эта функция обычно используется в двух ситуациях. В первом случае она позволяет заменить значение $null$ на некоторое реальное значение (например, на ноль для числовых данных или на пустую строку для символьных данных), что позволяет избежать ошибки при вычислениях. Во втором случае она позволяет заменить значение $null$, которое не может использоваться напрямую само по себе, на некоторое значение, которое заведомо не может встречаться среди рассматриваемых данных, и в

дальнейшем организовать проверки по этому значению. Типы значений для Value1 и Value2 могут быть любыми, но обязаны совпадать у обоих значений;

- **IFEMPTY**(Value1, Value2). Возвращает значение Value2, если значение Value1 равно null или пустой строке, иначе возвращает значение Value1;
- **ISNULL**(Value). Возвращает истину, если значение Value равно null;
- **ISDEFINED**(String). Возвращает истину, если переменная с указанным именем определена.

Строковые функции

Категория «Строковые» объединяет функции, работающие со строками. В нее входят следующие функции.

- **LENGTH**(String) – возвращает количество символов в указанной строке;
- **REPLACE**(Source, Pattern, NewString) – возвращает строку Source, в которой все вхождения подстроки Pattern заменены на подстроку NewString;
- **SUBSTR**(String, Index, Count) – возвращает часть строки String, начиная с символа с порядковым номером Index и количеством символов Count;
- **LEFT**(String, Count) – возвращает строку, состоящую из первых Count символов исходной строки;
- **RIGHT**(String, Count) – возвращает строку, состоящую из последних Count символов исходной строки;
- **POS**(Substring, String [, N]) – ищет N-е по счету вхождение подстроки Substring в строку String и возвращает порядковый номер первого найденного символа. Если параметр N не указан, он предполагается равным 1. Если подстрока не найдена, функция возвращает 0;
- **REPEATTEXT**(Text, Count) – возвращает строку, составленную из Count раз повторенной строки Text;
- **UPPERCASE**(String) – возвращает исходную строку, в которой все буквы заменены на прописные;
- **LOWERCASE**(String) – возвращает исходную строку, в которой все буквы заменены на строчные;
- **CAPITALIZE**(String) – возвращает исходную строку, в которой все слова начинаются с прописной буквы;
- **TRIM**(String) – возвращает строку с удаленными начальными и конечными пробелами и служебными символами;
- **LTRIM**(String) – возвращает строку с удаленными начальными пробелами и служебными символами;
- **RTRIM**(String) – возвращает строку с удаленными конечными пробелами и служебными символами;
- **PADLEFT**(String, Length) – возвращает строку, дополненную слева пробелами до длины Length;
- **PADRIGHT**(String, Length) – возвращает строку, дополненную справа пробелами до длины Length;
- **PADLEFTCHAR**(String, Length, Char) – возвращает строку, дополненную слева символами Char до длины Length;
- **PADRIGHTCHAR**(String, Length, Char) – возвращает строку, дополненную справа символами Char до длины Length;
- **DELNONDIGIT**(String) – возвращает строку, из которой удалены все символы, кроме цифр и знаков «+», «-», «.»;
- **REVERSE**(String) – возвращает строку с переставленными в обратном порядке символами;

- **POSCHARS**(CharsStrings, String [, N]) – возвращает позицию N-го вхождения любого из указанных в строке CharsStrings символов в строку String. Под вхождением считается появление в строке любого из указанных символов. Например, POSCHARS('+-*/', ' x=2*y+z+t+4', 3) вернет 8, то есть позицию третьего знака действия, а не 10 – позицию третьего знака «+». Если параметр N не указан, он предполагается равным 1. Если N-е вхождение символов не найдено, функция возвращает 0;
- **LIKE**(String, PatternString) – возвращает истину, если указанная строка совпадает с шаблоном, указанным в PatternString. В шаблоне допускается использование специальных символов «?» и «*», означающих соответственно любой произвольный символ и любое количество произвольных символов. Вместо символа «?» может использоваться также символ «_», а вместо символа «*» – символ «%»;
- **ONLYCHARS**(String, CharsString). Возвращает истину, если строка String содержит только символы из набора символов, содержащихся в строке CharsString;
- **REPLACECHARS**(String, CharsString [, NewChar]). Заменяет в строке String все символы из набора CharsString на символ NewChar (удаляет все символы из набора, если CharsString не указан).

Функции форматирования

Категория «Форматирование» объединяет функции, преобразующие данные из одного типа в другой. В нее входят следующие функции.

- **TIMETOSTR**(Time [, Format string]) – возвращает значение времени Time, преобразованное в строку заданного формата. Формат задается с помощью строки форматирования вторым аргументом. Если этот параметр не указан, значение времени преобразуется в строку согласно системным настройкам. Символы в строке форматирования заменяются на следующие значения.

h	– заменяется на значение часа без ведущего нуля (0-23);
hh	– заменяется на значение часа с ведущим нулем (00-23);
n	– заменяется на значение минут без ведущего нуля (0-59);
nn	– заменяется на значение минут с ведущим нулем (00-59);
s	– заменяется на значение секунд без ведущего нуля (0-59);
ss	– заменяется на значение секунд с ведущим нулем (00-59);
:	– заменяется на заданный в системных настройках разделитель времени;
'xx'/'xx'	– символы в одиночных или двойных кавычках отображаются без изменений.
- **DATETOSTR**(Date [, Format string]) – возвращает значение даты Date, преобразованное в строку заданного формата. Формат задается с помощью строки форматирования вторым аргументом. Если этот параметр не указан, значение даты преобразуется в строку согласно системным настройкам. Символы в строке форматирования заменяются на следующие значения.

d	– заменяется на значение дня без ведущего нуля (1-31);
dd	– заменяется на значение дня с ведущим нулем (01-31);
ddd	– заменяется на аббревиатуру дня недели (Пон-Вск);
dddd	– заменяется на название дня недели (Понедельник-Воскресенье);
m	– заменяется на значение месяца без ведущего нуля (1-12);
mm	– заменяется на значение месяца с ведущим нулем (01-12);
mmm	– заменяется на аббревиатуру названия месяца (Янв-Дек);

- mmmm – заменяется на название месяца (Январь-Декабрь);
yy – заменяется на значение года из двух цифр (00-99);
yyyy – заменяется на значение года из четырех цифр (0000-9999);
/ – заменяется на заданный в системных настройках разделитель даты;
'xx'/'xx" – символы в одиночных или двойных кавычках отображаются без изменений.
- **STRTOINT**(String) – преобразует строку в целое число;
 - **STRTOFLOAT**(String) – преобразует строку в вещественное число;
 - **CHAR**(Number) – возвращает символ с заданным кодом;
 - **ASC**(String) – возвращает код первого символа заданной строки;
 - **INTTOSTR**(Number) – возвращает строку, соответствующую данному целому числу;
 - **FLOATTOSTR**(Number [, Width [, Decimals]]) – возвращает строку, соответствующую данному вещественному числу. Если параметры Width и Decimals не указаны, преобразование выполняется стандартным образом. Иначе форматирует строку таким образом, что выводятся Decimals знаков после запятой и общая длина строки составляет не менее Width символов (производится дополнение строки пробелами слева до необходимой длины). Если указан только параметр Width, параметр Decimals предполагается равным нулю;
 - **STRTODATE**(String) – преобразует строку соответствующего вида в дату;
 - **STRTOTIME**(String) – преобразует строку соответствующего вида в значение времени.
 - **DATETODAYS**(Date) – преобразует дату в количество дней, прошедшее от начала календарного отчета до данной даты. Эта функция полезна, если требуется найти дату, отстоящую на некоторое количество дней от заданной. В этом случае дата преобразуется с помощью указанной функции в количество дней, затем к этому количеству прибавляется или отнимается нужное значение, и затем выполняется обратное преобразование в дату с помощью функции «DAYSTODATE»;
 - **DAYSTODATE**(X) – преобразует количество дней, прошедшее от начала календарного отчета, в дату;
 - **TIMETODAYS**(Time) – возвращает долю дня для указанного значения времени. Например, выражение TIMETODAYS('06:00') вернет значение 0.25;
 - **DAYSTOTIME**(X) – преобразует долю дня в значение времени. Например, выражение DAYSTOTIME(0.5) вернет значение времени, равное 12 часам дня.
 - **TOROMAN**(Int). Возвращает строку с римской записью заданного числа;
 - **TOARAB**(String). Переводит строку с римской записью числа в число;
 - **FILETOSTR**(FileName [,Encoding]). Возвращает содержимое текстового файла в виде строки;
 - **STRTOBASE64**(String). Конвертирует строку в строку BASE64;
 - **BASE64TOSTR**(String). Конвертирует строку BASE64 в обычную строку;
 - **BASE64TOBLOB**(String). Конвертирует строку BASE64 в двоичные данные;
 - **BLOBTOBASE64**(Value). Конвертирует двоичные данные в строку BASE64;
 - **FILETOBLOB**(FileName). Возвращает содержимое текстового файла в виде двоичных данных;
 - **BASE64TOFILE**(String, FileName). Сохраняет данные из строки BASE64 в файл FileName.

Функции даты и времени

Категория «Даты и времени» объединяет функции, работающие со значениями даты и времени. В нее входят следующие функции.

- **TIME**(Hour, Min, Sec) – возвращает значение времени для указанного значения часов (Hour), минут (Min) и секунд (Sec);
- **GETTIME**(0) – возвращает текущее системное время. Аргумент функции может быть произвольным;
- **YEAR**(Date) – возвращает значение года для заданной даты Date;
- **DATE**(Year, Month, Day) – возвращает значение даты для указанного значения года (Year), месяца (Month) и дня (Day);
- **GETDATE**(0) – возвращает текущую системную дату. Аргумент функции может быть произвольным;
- **DAY**(Date) – возвращает значение дня для заданной даты Date;
- **DAYOFWEEK**(Date) – возвращает значение порядкового номера дня недели (понедельник – 1, вторник – 2, ..., воскресенье – 7) для заданной даты Date;
- **MONTH**(Date) – возвращает значение месяца для заданной даты Date;
- **MINUTE**(Time) – возвращает значение минут для заданного значения времени Time;
- **SECOND**(Time) – возвращает значение секунд для заданного значения времени Time;
- **HOURL**(Time) – возвращает значение часов (в 24-х часовом формате) для заданного значения времени Time;
- **NOW**(0) – возвращает текущую системную дату и время. Аргумент функции может быть произвольным.

Групповые функции

Категория «Групповые» объединяет функции, обрабатывающие несколько значений и возвращающие некий суммарный результат. Максимальное число аргументов для каждой из этих функций – 10. В эту категорию входят следующие функции.

- **MAXN**(Value1 [, Value2 [, Value3 ...]]) – возвращает максимальное значение из указанных аргументов. Аргументы могут быть произвольного типа, но тип должен быть одинаковым для всех аргументов. Числа сравниваются обычным образом, строки сравниваются в алфавитном порядке, дата и время – в календарном порядке, для логических значений ложь считается меньше, чем истина;
- **MINN**(Value1 [, Value2 [, Value3 ...]]) – возвращает минимальное значение из указанных аргументов. Правила использования этой функции такие же, как и для функции MAXN;
- **AVERAGEN**(Value1 [, Value2 [, Value3 ...]]) – возвращает среднее арифметическое значение для указанных аргументов. Аргументы могут быть только числового типа.

Агрегатные функции SQL

Категория «Агрегатные SQL» объединяет специальные агрегатные функции SQL. В нее входят следующие функции.

- **COUNT**(TableField | 0) – возвращает количество неповторяющихся значений поля. 0 – возвращает количество записей в таблице или выборке;
- **SUM**(TableField) – возвращает сумму всех значений поля. Значения Null игнорируются;
- **AVG**(TableField) – возвращает среднее для всех значений поля. Значения Null игнорируются;

- **MIN**(TableField) – возвращает минимальное значение указанного поля;
- **MAX**(TableField) – возвращает максимальное значение указанного поля.

Списочные функции

Категория «Списочные» объединяет функции, работающие со списками. В нее входят следующие функции.

- **INLIST**(Element, StringList [,DelimiterChar]) – возвращает истину, если Element присутствует в списке StringList. Списком считается строка, состоящая из нескольких элементов, отделяемых друг от друга символом DelimiterChar. Одним из вариантов получения списка является вызов для сущности агрегатной функции СПИСОК, рассмотренной выше. Пробелы между элементами списка не допускаются. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTADD**(StringList1, StringList2 [,DelimiterChar]) – возвращает объединение двух списков StringList1 и StringList2. Итоговый список состоит из элементов, входящих или в первый, или во второй список. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTSUB**(StringList1, StringList2 [,DelimiterChar]) – возвращает разность двух списков StringList1 и StringList2. Итоговый список состоит из элементов, входящих в список StringList1 и не входящих в список StringList2. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTMUL**(StringList1, StringList2 [,DelimiterChar]) – возвращает пересечение двух списков StringList1 и StringList2. Итоговый список состоит из элементов, входящих и в первый, и во второй список. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTPACK**(StringList [,DelimiterChar]) – возвращает список, из которого удалены повторяющиеся значения. Каждый элемент в итоговом списке будет встречаться только один раз, из одинаковых элементов будет оставлен самый первый, так что порядок элементов в списке не нарушается. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTCONCAT**(StringList1, StringList2, ConcatDelimiter [,DelimiterChar]) – соединяет поэлементно два списка StringList1 и StringList2 таким образом, что итоговый список имеет вид «Сп1Эл1#Сп2Эл1;Сп1Эл2#Сп2Эл2;Сп1Эл3#Сп2Эл3...», где «#» - значение ConcatDelimiter. Готовый список может быть использован, например, в качестве параметра для системной сущности \$СПИСОК_N (описана ниже) или в качестве самостоятельного списка. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTCROSS**(StringList1, StringList2, ConcatDelimiter [,DelimiterChar]) – возвращает список, в котором каждый элемент списка StringList1 соединен с каждым элементом списка StringList2. Итоговый список имеет вид «Сп1Эл1#Сп2Эл1;Сп1Эл1#Сп2Эл2;Сп1Эл1#Сп2Эл3...;Сп1Эл2#Сп2Эл1;Сп1Эл2#Сп2Эл2...», где «#» - значение ConcatDelimiter. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **NUMLIST**(Start, End [,Step]) – возвращает список из целых чисел, начинающийся с числа Start и заканчивающийся числом End, с шагом Step (если параметр Step не задан, он предполагается равным 1);
- **LISTITEM**(StringList, Num [,DelimiterChar]) – возвращает элемент списка StringList с порядковым номером Num (элементы нумеруются, начиная с 1). Если параметр DelimiterChar не указан, он предполагается равным «;»;

- **LISTITEMN**(StringList, Num, ItemNum, ConcatDelimiter [, DelimiterChar]) – аналог функции LISTITEM для составного списка. Возвращает часть элемента составного списка с номером Num, номер части – ItemNum, разделитель частей – ConcatDelimiter. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTLEN**(StringList [, DelimiterChar]) – возвращает количество элементов в списке. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTINTERLACE**(StringList1, StringList2, StringList3, ЗначениеСравнения [, УсловиеСравнения] [, DelimiterChar]) – возвращает список, полученный объединением элементов списков StringList2 и StringList3. Элементы для объединенного списка выбираются следующим образом. Берется очередной элемент списка StringList1 и сравнивается со значением ЗначениеСравнения. Если параметр УсловиеСравнения не указан или равен нулю, то происходит сравнение элемента списка и ЗначениеСравнения на равенство (совпадение). Если параметр УсловиеСравнения больше нуля, то проверяется, не является ли элемент списка большим, чем ЗначениеСравнения, если параметр УсловиеСравнения меньше нуля, то проверяется, не является ли элемент списка меньшим, чем ЗначениеСравнения. Сравнение происходит в зависимости от типа параметра ЗначениеСравнения – если этот параметр является числом, то элементы списка StringList1 также преобразуются перед сравнением в числа и сравнение происходит как сравнение двух чисел. Если параметр ЗначениеСравнения является строкой, то сравниваются строки и т.д. То есть, каждый элемент списка StringList1 перед сравнением преобразуется в тип, соответствующий типу параметра ЗначениеСравнения. Если сравнение вернуло истину, то в качестве очередного элемента результирующего списка берется очередной элемент списка StringList2, если ложь – очередной элемент списка StringList3. Если параметр DelimiterChar не указан, он предполагается равным «;»;
- **LISTGROUPFILTER**(StringList, FilterList [, DelimiterChar]) – Возвращает список из элементов StringList (разделитель элементов по умолчанию ";"), в котором берутся элементы, соответствующие каждому первому из группы одинаковых элементов списка FilterList. Например, LISTGROUPFILTER('4;12;10;8;11;6;2;4', '1;1;1;2;2;3;3;3') вернет список '4;8;6';
- **LISTINDEX**(StringList [, DelimiterChar]). Возвращает строку индекса списка для дальнейшего использования при быстром обращении к элементам списка с помощью функции LISTITEMBYIDX;
- **LISTITEMBYIDX**(StringList, ListIndex, Num [, DelimiterChar]). Возвращает элемент списка с номером Num, используя индекс списка, полученный с помощью функции LISTINDEX (разделитель элементов по умолчанию ";");
- **LISTFIND**(StringList1, StringList2 [, IgnoreCase, DelimiterChar]). Возвращает список из позиций элементов списка StringList1 в списке StringList2 (разделитель элементов по умолчанию ";"). Если IgnoreCase установлен в false или в 0 (по умолчанию), сравнение ведется с учетом регистра, иначе - без учета;
- **LISTSET**(Value, StringList1, Num [, DelimiterChar]). Заменяет в списке элемент с номером Num на значение Value (разделитель элементов по умолчанию ";"). Возвращает измененный список;
- **LISTSUBALL**(StringList1, StringList2 [, DelimiterChar]). Возвращает разность списков StringList1 и StringList2, при этом в списке StringList1 удаляются все элементы, встречающиеся в StringList2, а не только первый, как для LISTSUB (разделитель элементов по умолчанию ";");

- **LISTFINDN**(StringList1, StringList2, ItemNum, ConcatDelimiter [,IgnoreCase, DelimiterChar]). Возвращает список из позиций элементов списка StringList1 в составном списке StringList2, в части с порядковым номером ItemNum, разделитель частей - ConcatDelimiter (разделитель элементов по умолчанию ";"). Если IgnoreCase установлен в false или в 0 (по умолчанию), сравнение ведется с учетом регистра, иначе – без учета;
- **LISTGROUPLIST**(StringList, FilterList, ConcatDelimiter [, DelimiterChar]). Возвращает список из элементов StringList (разделитель элементов по умолчанию ";"), в котором элементы, соответствующие каждому элементу из группы одинаковых элементов списка FilterList, объединяются во внутренний список с разделителем ConcatDelimiter;
- **VALLIST**(StringValue, Count [, DelimiterChar]). Возвращает список из Count копий значения StringValue (разделитель элементов по умолчанию ";");
- **LISTCUT**(StringList, Num, ConcatDelimiter [, DelimiterChar]). Убирает из каждого элемента составного списка (разделитель частей - ConcatDelimiter) часть с номером Num (разделитель элементов по умолчанию ";");
- **LISTCONCATNONEMPTY**(StringList1, StringList2, ConcatDelimiter [, DelimiterChar]). Возвращает список, в котором элементы списков StringList1 и StringList2 соединены поэлементно через разделитель ConcatDelimiter. Если один из элементов списка пустой, разделитель ConcatDelimiter для этой пары элементов не ставится.

Другие функции

Категория «Другие» содержит некоторые специальные функции, в том числе функции для работы со словарями и коллекциями.

- **SHA512HASH**(String). Возвращает для строки хэш-строку по алгоритму SHA512;
- **MAKEPROGRESSFORM**(Caption, MaxPos [, FormWidth, Show, ShiftX, ShiftY]). Создает форму для отображения прогресса операции, с заголовком Caption и числом позиций MaxPos. Ширина формы равна FormWidth (по умолчанию 0, что означает стандартную ширину). Если Show не равно 0 (1 по умолчанию), форма сразу же появляется на экране;
- **REMOVEPROGRESSFORM**(0). Прячет форму для отображения прогресса операции и освобождает ее;
- **SHOWPROGRESSFORM**(0). Показывает форму для отображения прогресса операции;
- **HIDEPROGRESSFORM**(0). Прячет форму для отображения прогресса операции;
- **SETPROGRESSFORM**(PositionInc, [, Caption]). Увеличивает позицию формы прогресса операции на PositionInc. Если Caption не пустая строка (по умолчанию), меняет заголовок формы на Caption;
- **SETVARIABLEMODE**(Mode). Устанавливает режим работы при обращении к неинициализированным переменным (0 - выдача ошибки, 1 - возврат значения null) ;
- **RUNREPORT**(ReportName, ReportMode, ParameterMode, VariableMode, Param1, Param2, ...). Запускает отчет из файла ReportName и возвращает результат в виде строки. ReportMode: 0=Excel, 1=HTML, 2=XML. ParameterMode (ввод пользователем значений параметров): 0=никогда, 1=если есть неопределенные параметры, 2=если есть неопределенные необходимые параметры, 3=всегда. VariableMode - режим работы при обращении к

- неинициализированным переменным (см. описание функции SETVARIABLEMODE). Param1, Param2... – значения параметров отчета;
- **CHECKXMLSCHEMA**(Xml, Schema). Проверяет соответствие файла Xml схеме Schema. Возвращает пустую строку в случае соответствия, иначе – строку с ошибками;
 - **DICTCREATE**(0). Создает новый словарь и возвращает его уникальное имя;
 - **DICTADD**(Name, Keys, Values [, DelimiterChar]). Добавляет данные в словарь с именем Name, используя список ключей Keys и список значений Values (разделитель элементов по умолчанию ";");
 - **DICTREMOVE**(Name, Keys [, DelimiterChar]). Удаляет ключи Keys из словаря с именем Name (разделитель элементов по умолчанию ";");
 - **DICTCLOSE**(Name). Освобождает словарь с именем Name;
 - **DICTGET**(Name, Key). Возвращает значение для ключа Key из словаря с именем Name. Если ключа нет в словаре, возвращает null;
 - **DICTCONTAINS**(Name, Key). Возвращает истину, если ключ Key есть в словаре с именем Name;
 - **COLLCREATE**(0). Создает новую коллекцию и возвращает ее уникальное имя;
 - **COLLADD**(Name, Values [, DelimiterChar]). Добавляет данные в коллекцию с именем Name, используя список значений Values (разделитель элементов по умолчанию ";");
 - **COLLREMOVE**(Name, Num). Удаляет элемент с порядковым номером Num из коллекции с именем Name;
 - **COLLCLOSE**(Name). Освобождает коллекцию с именем Name;
 - **COLLITEM**(Name, Num). Возвращает элемент с порядковым номером Num из коллекции с именем Name;
 - **COLLCOUNT**(Name). Возвращает количество элементов в коллекции с именем Name;
 - **COLLIST**(Name [, DelimiterChar]). Возвращает содержимое коллекции с именем Name в виде списка (разделитель элементов по умолчанию ";");
 - **FILEFIND**(Path [, DelimiterChar]). Возвращает список имен файлов в указанном каталоге (разделитель элементов по умолчанию ";");
 - **DICTKEYLIST**(Name [, DelimiterChar]). Возвращает содержимое ключей словаря с именем Name в виде списка (разделитель элементов по умолчанию ";");
 - **COLLSET**(Name, Value, Num). Заменяет в коллекции с именем Name элемент с номером Num на значение Value. Возвращает true при успешном выполнении;
 - **SPLITSTRINGSCREEN**(String, Font, Width [, ForceBreak, Hyphenation, DelimiterChar]). Разбивает строку на список строк таким образом, чтобы части строки при выводе шрифтом Font умещались в ширину Width. Font представляет собой строку, в которой через ";" перечислены имя шрифта, его размер и атрибуты (BIUS). Строка разбивается по пробелам. Если параметр ForceBreak=1 (по умолчанию), то часть строки, в которой нет пробелов и которая не умещается в заданную ширину, будет разбита принудительно по символам, иначе останется в неизменном виде. Если параметр Hyphenation=1 (по умолчанию), то при необходимости будет использоваться перенос слов по правилам русского языка. Разделитель элементов итогового списка по умолчанию ";";
 - **GETUNIQUETEMPFILENAME**(Prefix, [, Ext, Path]). Возвращает уникальное имя для временного файла в директории %TEMP%. Имя файла будет

начинаться с заданного префикса Prefix, файл будет иметь расширение Ext (по умолчанию – .tmp). Если задан путь Path, проверяется содержимое данной директории, а не директории %TEMP%.

Скрипты

Скрипт – сложное выражение, состоящее из нескольких формул. Скрипт должен начинаться с ключевого слова *&script*. Каждая новая формула скрипта должна начинаться с символа *&*. Результатом скрипта является результат последней формулы этого скрипта.

В скриптах можно использовать следующие конструкции.

- **Локальные переменные.** Присваиваются просто знаком равенства: *Локальная переменная = <Выражение>*. Например,

```
&script
&x=1
```

- **Формулы, написанные на языке построения отчетов.** В том числе, все встроенные и агрегатные функции, параметры и переменные. Например,

```
&фио=#СЗ_С_ИСПОЛНИТЕЛЬ.ФИО(Фильтр='№_П_П=x')
```

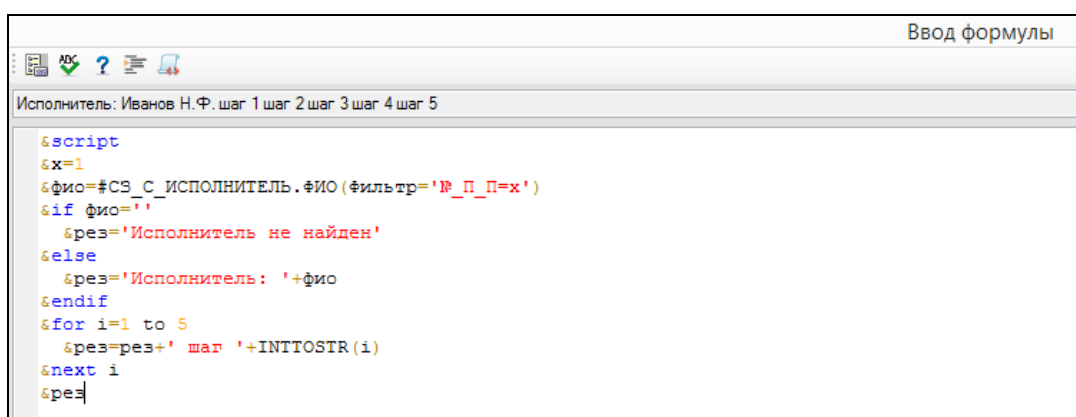
- **Условный оператор.** Ключевые слова: *if, elseif, else, endif*. Например,

```
&if фио=""
&рез='Исполнитель не найден'
&else
&рез='Исполнитель: '+фио
&endif
```

- **Циклы.** Ключевые слова: *for, to, next; while, wend; repeat, until*. Например,

```
&for i=1 to 5
&рез=рез+' шаг '+INTTOSTR(i)
&next i
```

Ниже приведен пример скрипта в форме ввода формул.



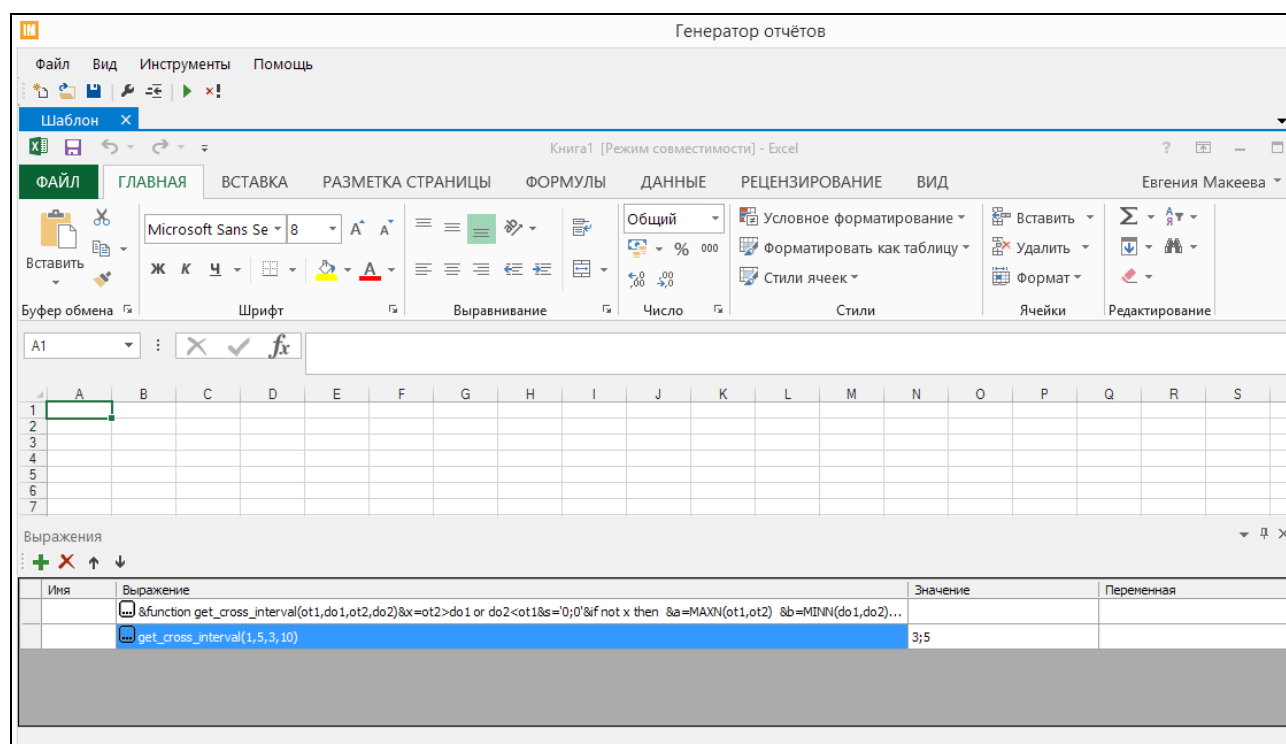
Пользовательские функции

В некоторых случаях встроенных функций генератора отчетов может оказаться недостаточно, и автор отчета имеет возможность добавить в отчет собственные специальные

функции и потом использовать их при написании выражений. Функция должна быть объявлена до ее использования, поэтому обычно функции пишут в первых выражениях отчета до начала всех вычислений. Оформляются функции подобно [скриптам](#), можно использовать все те же конструкции языка для написания формул. Функция должна начинаться с ключевого слова *&function* <имя функции>. Каждая новая формула функции должна начинаться с символа *&*. Результатом функции является результат последней формулы этой функции. Функция может иметь параметры, которые перечисляются в скобках после имени функции, через запятую. Ниже приведен пример функции `get_cross_interval`, которой передаются два интервала `ot1-do1` и `ot2-do2` и требуется найти их пересечение (интервал пересечения является результатом).

```
&function get_cross_interval(ot1,do1,ot2,do2)
&x=ot2>do1 or do2<ot1
&s='0;0'
&if not x then
  &a=MAXN(ot1,ot2)
  &b=MINN(do1,do2)
  &s=FLOATTOSTR(a)+';'+FLOATTOSTR(b)
&endif
&s
```

Далее функцию можно вызывать в любом выражении отчета. Например, вызов `get_cross_interval(1,5,3,10)` вернет результат 3;5.



Дополнительные элементы языка

Дополнительные параметры при получении свойства сущности

Каждое обращение к свойству какой-либо сущности (в простом варианте) выглядит следующим образом:

#<Сущность>.<Свойство>(<параметры отбора>)

Параметры отбора для записей сущности делятся на два вида. Для каждой сущности, во-первых, есть свои варианты параметров отбора, причем количество таких вариантов и параметры отбора, используемые в них, строго заданы для каждой конкретной сущности и не могут меняться. Но помимо этого, есть еще дополнительные параметры отбора, которые могут быть указаны в любом выражении отбора для любой сущности, помимо самих параметров сущности. Выше уже были рассмотрены два таких параметра – это параметр «Фильтр», позволяющий добавить произвольный критерий отбора записей, дополнительно к основному, и параметр «Цикл», с помощью которого можно указать цикл, к которому относится данный набор записей (в основном, это необходимо при пересечении циклов). Кроме этого, могут использоваться и другие дополнительные параметры. Ниже приведен их полный список с описанием назначения каждого параметра.

Параметр «Фильтр»

Форма записи:

Фильтр=<строка фильтра>

Позволяет задать в символьном виде дополнительные условия отбора, которым должны удовлетворять записи сущности. Строка фильтра представляет собой логическое выражение, в котором используются поля текущей сущности. Теоретически, это может быть любое логическое выражение генератора отчетов, в том числе и обращающееся к другим сущностям, но на практике из соображений быстродействия подобные проверки лучше выполнять с использованием списков, а в выражении фильтра использовать только поля из текущей сущности. В выражении фильтра допускается использование встроенных функций и переменных отчета.

Ниже приведены некоторые примеры выражений фильтра:

```
#СЗ_ФИЗИЧЕСКОЕ_ЛИЦО.ID_ФИЗИЧЕСКОЕ_ЛИЦО(Фильтр='ISNULL(КОНТАКТНЫЙ_ТЕЛЕФОН) or  
ISNULL(АДРЕС_ЭЛЕКТРОННОЙ_ПОЧТЫ)')
```

```
#СЗ_ЗАМЕЧАНИЕ.ID_ЗАМЕЧАНИЕ.СПИСОК(Фильтр='ДАТА_ВЫПОЛНЕНИЯ>=:C and  
ДАТА_ВЫПОЛНЕНИЯ<=:ПО')
```

```
#НАСТРОЙКА_ОБМЕНА.НАИМЕНОВАНИЕ_ПОЛНОЕ.СПИСОК(Фильтр='INLIST(НОМ_ПП,сп_ном) and  
ID_ИСТОЧНИК_ДАННЫХ_ФОРМАТ=1')
```

Параметр «Цикл»

Форма записи:

Цикл=<строка с именем цикла>

Данный параметр позволяет явно задать имя цикла, размер которого должен зависеть от количества записей в сущности при данном отборе. Обычно этот параметр используется при пересечении горизонтальных и вертикальных циклов, когда генератор отчетов не может однозначно распознать, к какому циклу относится выражение отбора записей из сущности в ячейке пересечения циклов. Кроме того, иногда бывают ситуации, когда записи из какой-либо сущности не нужно учитывать при подсчете размера цикла. В этом случае в выражении отбора нужно указать параметр «Цикл» и в качестве имени цикла подставить специальную системную переменную \$NONE («Цикл=\$NONE»).

Параметр «Запись»

Форма записи:

Запись=<номер записи>

С помощью этого параметра можно выбрать одну конкретную запись из отобранных записей сущности, с указанным порядковым номером. Если номер записи превышает количество отобранных записей, берется последняя запись. Например, выражение

```
#СЗ_С_ИСПОЛНИТЕЛЬ.ФИО(Фильтр='LEFT(UPPERCASE(ФИО),1)="И"')
```

возвращает наименование ФИО одного из исполнителей, начинающееся с символа «И». Если указать для отбора данных параметр «Запись», например так:

```
#СЗ_С_ИСПОЛНИТЕЛЬ.ФИО(Фильтр='LEFT(UPPERCASE(ФИО),1)="И"', Запись=4)
```

то формула выдаст ФИО исполнителя, начинающееся с символа «И» и идущее четвертым по порядку в наборе данных. Надо отметить, что поскольку порядок записей при отборе данных неопределен, то при использовании последнего выражения будет получена единственная запись, но какая именно – сказать невозможно, и более того, при разных обращениях к выражению теоретически можно для каждого обращения получать разные значения. Поэтому, как правило, параметр «Запись» имеет смысл только в комбинации с другими параметрами, такими как «Сортировка» или «Привязка», которые будут рассмотрены ниже.

Параметр «Сортировка»

Форма записи:

Сортировка=<Имя свойства сущности для сортировки>

Данный параметр при отборе данных возвращает набор записей, отсортированный по указанному свойству. Имя свойства должно быть задано в виде символьной строки. Например, в предыдущей формуле возможно добавить этот параметр следующим образом:

```
#СЗ_С_ИСПОЛНИТЕЛЬ.ФИО(Фильтр='LEFT(UPPERCASE(ФИО),1)="И"', Запись=4, Сортировка='ФИО')
```

С помощью этого выражения мы всегда будем получать одно и то же значение, являющееся ФИО исполнителя, начинающимся с символа «И» и идущим четвертым среди таких исполнителей в наборе данных в алфавитном порядке.

Этот параметр очень часто применяется при использовании агрегатной функции «Список». Обычно он имеет значение свойства ID в сущности или другого свойства сущности с уникальными значениями. Например, выражение:

```
#НАСТРОЙКА_ОБМЕНА.НАИМЕНОВАНИЕ_ПОЛНОЕ.СПИСОК()
```

вернет список всех наименований настроек обмена. Эти наименования будут идти в произвольном порядке, и более того, при разных запусках генератора отчетов этот порядок может отличаться.

Если задать еще одно выражение:

```
#НАСТРОЙКА_ОБМЕНА.НОМ_ПП.СПИСОК()
```

Это выражение вернет список порядковых номеров настроек обмена. Порядок номеров будет также произвольным.

Если сохранить значение первого выражения (со списком наименований) в переменную *lst1*, а второго выражения (со списком кодов) – в переменную *lst2*, то можно задать следующее выражение:

```
LISTCONCAT(lst1, lst2, '#')
```

Это выражение объединит два списка *lst1* и *lst2* в один список из объединенных элементов (наименования и порядкового номера настройки обмена, через символ «#»), примерно такого вида: «Экспорт настроек обмена#1;Импорт настроек обмена#2;Импорт Заказчиков и проектов#10;Экспорт заданий#11;Импорт заданий#12;Импорт физических лиц#16;Экспорт запросов к БД#17;Импорт запросов к БД#18;...».

Для этого списка ожидается, что каждому наименованию настройки обмена будет соответствовать ее порядковый номер, т.е. настройка экспорта настроек обмена будет иметь номер 1, настройка экспорта заданий будет иметь номер 11, настройка импорта запросов к БД – номер 18 и т.д. Но на самом деле это может быть не так, и с каким-либо наименованием настройки будет связан номер совершенно от другой настройки! Действительно, поскольку наименования и коды для каждого из выражений возвращаются в произвольном порядке, нет никаких гарантий, что этот порядок будет одинаковым для выражений *lst1* и *lst2*. Чтобы гарантировать, что каждому наименованию настройки будет сопоставлен именно ее номер, необходимо использовать в обеих формулах параметр «Сортировка» по одному и тому же свойству сущности с уникальными значениями, например, по свойству ID. Т.е. выражения для *lst1* и *lst2* должны выглядеть так:

```
#НАСТРОЙКА_ОБМЕНА.НАИМЕНОВАНИЕ_ПОЛНОЕ.СПИСОК(Сортировка='ID_НАСТРОЙКА_ОБМЕНА')  
#НАСТРОЙКА_ОБМЕНА.НОМ_ПП.СПИСОК(Сортировка='ID_НАСТРОЙКА_ОБМЕНА')
```

Теперь выражение для объединения списков вернет значение списка из комбинаций наименований и порядковых номеров настроек обмена, в котором гарантированно каждому наименованию настройки обмена будет соответствовать номер этой же настройки. Следует обратить внимание, что свойство сущности для сортировки должно иметь уникальные значения в сущности, иначе же для записей с одинаковым значением свойства взаимный порядок этих записей будет опять неопределенным.

Такие составные списки из элементов нескольких списков достаточно часто применяются в отчетах, обычно в комбинации с системными сущностями «\$СПИСОК» и «\$СПИСОК_N», которые подробно будут рассмотрены [ниже](#).

Параметр «Привязка»

Форма записи:

Привязка=<имя свойства для привязки>;<список привязки>

Значение данного параметра представляет собой строку из двух частей, разделяемых символом «;». Первая часть представляет собой наименование свойства текущей сущности, вторая часть – выражение (обычно имя переменной) для некоторого списка. Использование этого параметра позволяет отобрать из сущности записи, количество и порядок следования которых будут соответствовать порядку элементов в списке привязки, причем для связи элементов списка и записей сущности используются значения свойства для привязки.

Ниже пояснено использование этого параметра на основе предыдущего примера. Предположим, что нужно поставить в соответствие каждому наименованию настройки обмена наименование формата обмена, соответствующего этой настройке. В сущности

«#НАСТРОЙКА_ОБМЕНА» есть свойство «ID_ИСТОЧНИК_ДАННЫХ_ФОРМАТ», но как получить список наименований форматов данных, соответствующий каждой настройке обмена?

Для этого понадобятся две формулы. Сначала нужно получить список ID форматов обмена, с элементами в том же порядке, что и в списках lst1 и lst2. Для этого можно применить следующую формулу:

```
#НАСТРОЙКА_ОБМЕНА.ID_ИСТОЧНИК_ДАННЫХ_ФОРМАТ.СПИСОК(Сортировка='ID_НАСТРОЙКА_ОБМЕНА')
```

Поскольку указан тот же набор параметров и сортировка, что и для списков lst1 и lst2, формула вернет список, согласованный с предыдущими. Если обозначить его переменной lst3, то можно задать следующую формулу:

```
#ФОРМАТ_ДАННЫХ_ОБМЕНА.НАИМЕНОВАНИЕ.СПИСОК(Привязка='ID_ФОРМАТ_ДАННЫХ_ОБМЕНА;lst3')
```

В результате будет получен список наименований форматов обмена, согласованный со списками lst1, lst2 и lst3. В качестве свойства привязки указано свойство ID из сущности форматов обмена и указано, что привязка должна быть сделана согласно списку lst3. Это означает, что первой записью из сущности #ФОРМАТ_ДАННЫХ_ОБМЕНА будет запись, у которой ID совпадает с первым элементом списка lst3, второй записью – запись с ID, совпадающим со вторым элементом списка lst3, и т.д. Если элементы в списке привязки lst3 повторяются, то будет несколько раз взята одна и та же запись из сущности #ФОРМАТ_ДАННЫХ_ОБМЕНА. На основе этого набора данных и будет составлен список наименований форматов обмена.

Если обозначить этот список с помощью переменной lst4, то можно задать формулу:

```
LISTCONCAT(lst1, lst4, '#')
```

Формула вернет список из объединенных элементов наименования настройки обмена и соответствующего ей наименования формата обмена, через символ «#», примерно такого вида: «Экспорт настроек обмена#Формат обмена ini;Импорт настроек обмена#Формат обмена ini;Импорт Заказчиков и проектов#Формат обмена таблица Excel;Экспорт заданий#Формат обмена ini;Импорт заданий#Формат обмена ini;Импорт физических лиц#Формат обмена таблица Excel;Экспорт запросов к БД#Формат обмена ini;Импорт запросов к БД#Формат обмена ini; ...»

Стоит отметить, что свойство для привязки должно иметь уникальные значения в своей сущности, иначе при привязке некоторым элементам списка будут сопоставлены произвольные записи сущности из тех, в которых значение поля привязки совпадает с элементом списка.

Параметр «Фильтр_списка»

Форма записи:

Фильтр_списка=<выражение для фильтра>;<список фильтрации>

Этот параметр представляет собою более мощную версию параметра привязки. С его помощью можно отобрать из сущности записи, количество и порядок следования которых будут соответствовать порядку элементов в списке фильтрации. При этом для связи элементов списка и записей сущности используется выражение для фильтра. В отличие от параметра привязки, использование выражения для связи позволяет задавать не только лишь

равенство для свойства сущности и элемента списка, а произвольное выражение над ними, и кроме этого, допускается использовать составные списки. Для каждого элемента списка фильтрации отбирается запись из сущности, для которой выражение для фильтра будет истинным. Если таких записей окажется несколько, будет взята произвольная из них (обычно такого быть не должно и выражение для фильтра должно быть задано так, чтобы оно было истинным не более чем для одной записи из текущей сущности). Если записей, для которых истинно выражение для фильтра, для какого-либо элемента списка фильтрации не окажется, для этого элемента будет взята «пустая» запись (запись, у которой все поля равны «null»).

В выражении для фильтра могут использоваться имена свойств текущей сущности и ссылки на текущий элемент списка фильтрации. Для ссылки на текущий элемент списка фильтрации используется специальное имя вида «\$\$эл<n>». Если список не составной, т.е. каждый его элемент не состоит из нескольких частей, то n всегда равно 1, т.е. обращение к текущему элементу списка фильтрации производится по имени \$\$эл1. Для составных списков n равно порядковому номеру части элемента.

Для примера будет использована формула из предыдущего примера, где объяснялся параметр «Привязка»:

```
#ФОРМАТ_ДАННЫХ_ОБМЕНА.НАИМЕНОВАНИЕ.СПИСОК(Привязка='ID_ФОРМАТ_ДАННЫХ_ОБМЕНА;lst3')
```

С использованием параметра «Фильтр_списка» эту же формулу можно записать следующим образом:

```
#ФОРМАТ_ДАННЫХ_ОБМЕНА.НАИМЕНОВАНИЕ.СПИСОК(Фильтр_списка='ID_ФОРМАТ_ДАННЫХ_ОБМЕНА=$$эл1;lst3')
```

Необходимо заметить, что для вычисления выражения для фильтра берутся не все записи в сущности, а только те из них, которые удовлетворяют остальным условиям отбора. То есть, если заданы какие-то условия отбора для конкретной сущности и (или) задан фильтр для этой сущности, то сначала к сущности применяются условия отбора и условия фильтра, а затем уже для оставшихся в результате записей происходит вычисление выражения для фильтра. Исключение составляют параметры «Сортировка» и «Запись» – эти параметры применяются уже после отработки фильтра списка.

Параметр «Пустое_Поле»

Форма записи:

Пустое_Поле=<значение для «пустого» элемента списка>

Данный параметр имеет смысл только при использовании агрегатной функции «СПИСОК». Значением параметра должна являться строка символов. Этот параметр указывает, что нужно подставить в качестве элемента списка, если поле для списка имеет в текущей записи пустое значение.

Параметр «Формат_даты»

Форма записи:

Формат_даты=<строка формата для даты/времени>

Данный параметр также имеет смысл только при использовании агрегатной функции «СПИСОК», причем только для случая, когда свойство сущности имеет тип даты и времени. Значением параметра должна являться строка формата для даты и/или времени. Этот

параметр указывает, в каком виде должна быть представлена дата и время при составлении списка.

Параметр «Группировка»

Форма записи:

Группировка=<выражение группировки>;<выражение значения>

Данный параметр применяется при необходимости посчитать значение какой-либо агрегатной функции не для всего набора данных, а отдельно по каждой группе данных. Данные делятся на группы с помощью задания выражения группировки. Это выражение содержит ссылки на свойства текущей сущности и считается для каждой записи из сущности (с учетом условий отбора и фильтрации, заданных другими параметрами). Записи с одинаковыми значениями этого выражения объединяются в группы, и для всех записей внутри каждой группы подсчитывается выражение значения. Выражение значения должно представлять собой одну из агрегатных функций (СУММА, КОЛВО и т.д.), после которой в скобках указано некоторое выражение над полями сущности. Это выражение считается для каждой записи из группы, и затем к этим выражениям применяется указанная агрегатная функция. При использовании данного параметра результирующий набор данных имеет не свойства текущей сущности, а два свойства с фиксированными именами: «ГРУППА» и «ЗНАЧЕНИЕ». Первое свойство – это значение выражения группировки для каждой группы, второе – значение выражения группировки для этой группы. Например, выражение

#СЗ_ДЕТАЛИЗАЦИЯ_ЗАМЕЧАНИЯ.ГРУППА.СПИСОК(Группировка='ID_ЗАМЕЧАНИЕ;КОЛВО(ID_ДЕТАЛИЗАЦИЯ)')

вернет список ID различных замечаний, для которых существуют записи детализации, а выражение

#СЗ_ДЕТАЛИЗАЦИЯ_ЗАМЕЧАНИЯ.ЗНАЧЕНИЕ.СПИСОК(Группировка='ID_ЗАМЕЧАНИЕ;КОЛВО(ID_ДЕТАЛИЗАЦИЯ)')

вернет для каждого из этих замечаний количество относящихся к нему записей детализации.

Системные сущности

Помимо задаваемых предметной областью сущностей, в генераторе отчетов существует несколько системных сущностей – «\$СПИСОК», «\$СПИСОК_N», «\$СПИСОК_P», «\$СПИСОК_N_P», предназначенных для более широкой поддержки работы со списками.

С помощью этих системных сущностей возможно представить любой список в виде полноценной сущности, к которой можно применить все дополнительные параметры отбора (фильтр, сортировку и др.) и организовать, например, с ее помощью циклы в отчете.

Обязательным параметром для системной сущности «\$СПИСОК» является параметр «СПИСОК», значением которого является список из элементов, разделенных «;». Результатом будет сущность с единственным свойством «ЭЛЕМЕНТ», значениями которого будут элементы списка. Это свойство всегда имеет символьный тип, поэтому если идет работа, например, со списком ID, для использования элемента в качестве значения для параметра в других выборках, может потребоваться его преобразование в численный вид, например, с помощью встроенной функции «STRTOINT».

Системная сущность «**\$СПИСОК_N**» используется для преобразования в сущность списков из комбинированных элементов (например, созданных с помощью встроенной функции «**LISTCONCAT**»). Для этой сущности добавляется еще один обязательный параметр «**РАЗДЕЛИТЕЛЬ**», в котором нужно указать, что является разделителем в составном элементе списка. Количество свойств в этой сущности определяется количеством частей в составном элементе списка, и эти свойства называются «**ЭЛЕМЕНТ_1**», «**ЭЛЕМЕНТ_2**», «**ЭЛЕМЕНТ_3**» и т.д. Эти свойства также имеют всегда символьный тип.

Системные сущности «**\$СПИСОК_P**» и «**\$СПИСОК_N_P**» эквивалентны сущностям «**\$СПИСОК**» и «**\$СПИСОК_N**», за исключением того, что разделитель элементов списка в них задается явно, а не равен «;».

Системные функции

Системные функции – это специальные функции, которые работают с некоторыми внутренними свойствами отчета и генератора отчетов. В окне ввода формул эти функции описаны для объекта формул «Функции» в специальной категории «Системные». Ниже приведен список этих функций с кратким описанием каждой из них.

- **\$СТИЛЬ(СПИСОК_ДИАПАЗОНОВ)** – возвращает строку описания стиля для списка диапазонов на листе отчета. Эта строка в специальном виде, описывающая ширину и оформление указанных ячеек, и в основном предназначена для передачи в качестве параметра для некоторых специальных сущностей на основе методов. Теоретически, может использоваться и пользователями для разбиения данных в соответствии с оформлением отчета;
- **\$ФОРМУЛА(Строка_формулы)** – позволяет вычислить формулу, заданную динамически, в виде символьной строки. То есть, формула не жестко задана в ячейке на листе настройки, а может меняться в зависимости от условий или данных из базы;
- **\$КЛАСС_В_СУЩН(ИмяКласса)** – возвращает для внутреннего программного класса данных с заданным именем пользовательское имя соответствующей сущности в генераторе отчетов, связанной с этим классом;
- **\$СУЩН_В_КЛАСС(ИмяСущности)** – по заданному пользовательскому имени сущности в генераторе отчетов возвращает имя внутреннего программного класса данных, связанного с этой сущностью;
- **\$ТАБЛ_В_СУЩН(ИмяТаблицы)** – возвращает для таблицы в базе данных с заданным именем пользовательское имя соответствующей этой таблице сущности в генераторе отчетов;
- **\$СУЩН_В_ТАБЛ(ИмяСущности)** – по заданному пользовательскому имени сущности в генераторе отчетов возвращает имя таблицы в базе данных, соответствующей этой сущности;
- **\$ПОЛЕ_В_СВОЙСТВО(ИмяСущности, ИмяПоля)** – возвращает для сущности с заданным пользовательским именем и имени поля в таблице базы данных (предполагается, что таблица в базе данных – это таблица, связанная с данной сущностью) имя соответствующего этому полю свойства сущности;
- **\$СВОЙСТВО_В_ПОЛЕ(ИмяСущности, ИмяСвойства)** – по заданному пользовательскому имени сущности в генераторе отчетов и по заданному пользовательскому имени свойства в этой сущности возвращает соответствующее этому свойству имя поля в таблице базы данных;
- **\$КЛАСССУЩН_В_СУЩН(ИмяКласса)** – возвращает для внутреннего программного класса-сущности с заданным именем пользовательское имя соответствующей сущности в генераторе отчетов, связанной с этим классом;

- **\$СУЩ_В_КЛАСССУЩ**(ИмяСущности) – по заданному пользовательскому имени сущности в генераторе отчетов возвращает имя внутреннего программного класса-сущности, связанного с этой сущностью;
- **\$LISTEXPR**(Expression, Start, End [,Step]) – возвращает список из значений выражения Expression. Внутри выражения используется внутренняя переменная \$\$CSTEP, которая последовательно изменяется от значения Start до значения End с шагом Step (если параметр Step не указан, он считается равным 1). На каждом таком шаге значение переменной \$\$CSTEP подставляется в выражение Expression, и значение этого выражения становится очередным элементом списка;
- **\$AGGREGATE**(Выражение,ИмяТаблицы [,Условие]). Возвращает в виде строки агрегатное выражение для заданной таблицы с фильтрацией по заданному условию;
- **\$SCHEMA**(0). Возвращает текущую схему БД;
- **\$DEBUG**(EXPR). Отображает значение выражения Expr;
- **\$ENTITY_TO_DATA**(ИмяКлассаСущности). Возвращает имя дата-класса по имени класса сущности;
- **\$ENTITY_TO_TABLE**(ИмяКлассаСущности). Возвращает имя таблицы по имени класса сущности;
- **\$DEBUGIF**(Condition, EXPR). Отображает значение выражения Expr, если истинно значение Condition;
- **\$MESSAGE**(Text). Отображает значение Text;
- **\$MESSAGEIF**(Condition, Text). Отображает значение Text, если истинно значение Condition;
- **\$CALLMETHOD**(ClassName, MethodName, Param1, Param2, Param3...). Запускает метод класса и возвращает результат в виде строки;
- **\$GETSEQUENCENEXT**(SeqName). Возвращает очередной элемент последовательности с заданным именем;
- **\$SHOWTABLE**(TableData, Header, TableName, ColumnDefs, EnttList, VisibleColumns, FieldDelimiter, RecDelimiter). Отображает содержимое таблицы;
- **\$APPPATH**(0). Возвращает путь текущего приложения;
- **\$PURE**(String). Псевдофункция для фильтрации;
- **\$СПИСОК_СВОЙСТВ**(ИмяСущности). Возвращает для сущности с заданным именем список имен свойств данной сущности;
- **\$ENTITYSELECT**(ИмяСущности [,MultiSelect,UserCondition,Title]). Вызывает для сущности с заданным именем форму выбора и позволяет выбрать одну или несколько записей из сущности (возвращается значение или список значений ID);
- **\$CLEARBUFFER**(0). Очищает буфер базы данных (для увеличения объема доступной памяти);
- **\$TOLOG**. Запись в консоль лога;
- **\$CREATEOUTERTABLEFROMLIST**(DataList, FieldNameList [, ConcatDelimiter [, DelimiterChar]]). Формирует внешнюю таблицу с именами полей из списка FieldNameList и данными из составного списка DataList. Если разделитель записей ConcatDelimiter не указан, таблица будет состоять из одного поля с именем FieldNameList и список DataList считается простым. Возвращает список из имени файла таблицы и запроса для обращения к таблице;
- **\$DELETEOUTERTABLE**(TableName). Удаляет внешнюю таблицу с именем TableName.

Завершающий макрос

Для некоторых отчетов сложной структуры стандартных средств генератора отчетов может оказаться недостаточно. Например, может потребоваться какое-то особое оформление для данных, выходящих за определенные границы, или нужно удалить из готового отчета столбцы или строки с нулевыми или повторяющимися данными. Для этих целей можно написать в шаблоне отчета макрос на языке Visual Basic для Excel, который запустится автоматически после завершения формирования отчета и до его показа на экране. Этот макрос должен иметь имя «**EndReport**». Внутри него можно разместить любой код на языке макросов, который нужным образом преобразует сформированный генератором отчет.

Специальные символы и эффекты

Для оформления отчетов могут быть использованы специальные символы и эффекты. Они задаются с помощью специальных символьных последовательностей, которые записываются непосредственно в содержимое ячеек на листе отчета (как напрямую, в шаблоне, так и в качестве результата вычисления выражений) и перед показом отчета на экране преобразуются в соответствующие элементы оформления. В текущей версии доступен один специальный эффект и 4 специальных символа (в последующих версиях их количество может быть увеличено в соответствии с потребностями пользователей).

Специальные эффекты задаются в виде последовательностей символов следующего вида:

##Имя=<имя эффекта>;X1=<значение параметра>;X2=<значение параметра>;...##

Каждый эффект задается своим именем, количество и значения параметров зависят от вида эффекта. Если внутри значения параметра необходимо указать символ «;», необходимо перед и после значения параметра поставить символ «"».

В генераторе отчетов доступен специальный эффект с именем «Индекс». Этот эффект позволяет выводить текст, справа от которого могут находиться символы в верхнем и нижнем индексе, выводимые на одном уровне по вертикали. Для него определено три параметра. Первый параметр задает базовый текст, который будет выведен в ячейке обычным, текущим для ячейки оформлением. Второй параметр задает текст, который будет выведен после базового текста и оформлен верхним индексом. Третий параметр задает текст, который будет выведен после базового текста и оформлен нижним индексом, причем вертикальный уровень символов верхнего и нижнего индекса будет совпадать. Например, выражение «Точить фаску на диаметр ##Имя=Индекс;X1=87;X2=-0,2;X3=-0,3## заданным инструментом» будет выведено в ячейке отчета в следующем виде:

Точить фаску на диаметр 87 ^{-0,2} _{-0,3} заданным инструментом
--

Выражение «возьмем ##Имя=Индекс;X1=X;X2="карб.;кисл.";X3=2## в тройном количестве» будет выглядеть в отчете так:

возьмем X ₂ ^{карб.;кисл.} в тройном количестве
--

Специальные символы задаются в виде последовательностей символов следующего вида:

&<имя символа>;

В генераторе отчетов определены следующие специальные символы:

diam – знак диаметра;	∅
angle – знак угла;	∠
dgr – знак градуса;	°
plmin – знак «плюс-минус».	±

Например, выражение «сверло ⋄ 5 мм.» будет выглядеть в отчете так:

сверло ∅ 5 мм.

Создание нового отчета

Создание шаблона

Для того, чтобы создать новый шаблон отчета, необходимо открыть генератор отчетов и нажать кнопку «Новый». Программа создаст новый начальный шаблон отчета и загрузит его в генератор.

Физически шаблон отчета представляет собой два файла. Первый файл формата *MS Excel* содержит непосредственно видимую часть отчета, т.е. некоторую форму, которая заполняется данными и выводится для просмотра на экран или на печать. Данные для этой формы рассчитываются на основе выражений, хранящихся во втором файле формата *rprm*.

Самая общая, упрощенная схема представления отчета может быть описана следующим образом. В файле формата *rprm* сохраняются некоторые выражения, написанные на языке построения отчетов. Каждое из этих выражений при расчете возвращает некоторый результат. В файле формата *MS Excel* в нужных местах указываются ссылки на результат расчета выражений из файла формата *rprm*. В результате файл формата *MS Excel* после расчета оказывается заполнен рассчитанными данными.

Сохранение шаблона

Для сохранения изменений в шаблоне отчета нужно нажать кнопку «Сохранить». При первом сохранении будет запрошено имя файла шаблона, которое нужно ввести. В дальнейшем можно периодически сохранять изменения в шаблоне с помощью этой же кнопки.

Общий порядок создания отчета

После того, как шаблон нового отчета создан и сохранен, следует определить параметры отчета (см. главу [Параметры отчета](#)), т.к. именно от входных параметров зависит полученный результат. Обычно в отчетах в качестве параметров используются внутренние коды (ID) того или иного объекта (но это не обязательно). Если в конкретном отчете параметры не нужны, то их можно не задавать.

Существует несколько путей создания отчета. Можно сначала создать основную форму для вывода на [панели MS Excel](#), а затем уже написать необходимые выражения на [панели выражений](#) и установить на панели MS Excel ссылки на соответствующие *Имена* выражений. Можно поступить наоборот – сначала написать все необходимые выражения, а уже затем создавать видимую форму отчета, подставляя в процессе создания ссылки на необходимые *Имена* выражений. Наконец, можно комбинировать оба этих подхода, в

произвольном порядке заполняя форму отчета в MS Excel, вводя выражения и устанавливая связи между выражениями и панелью MS Excel.

При необходимости (не обязательно) нужно определить циклы отчета на [панели циклов](#). Это можно делать как в последнюю очередь, так и в процессе написания выражений отчета.

Формирование и просмотр отчета

Готовый отчет или любой промежуточный вариант в целях отладки можно запустить на выполнение непосредственно из генератора отчетов. Для формирования отчета для текущих значений параметров, нужно нажать на кнопку «Запуск». На экране появится отдельное окно MS Excel со сформированным отчетом.

Приложение 1. Привилегии для работы с модулем «Генератор отчетов»

Привилегия	Назначение	Использование	Входит в комплект поставки
Call_Interpreter_Formula	Генератор отчетов	Модуль "Генератор отчетов"	Standard, Extended, Full

Приложение 2. Действия по клавишам

Приведены стандартные действия по клавишам и комбинациям клавиш в форме генератора отчетов.

- Ctrl+N – создать новый шаблон.
- Ctrl+O – открыть шаблон.
- Ctrl+S – сохранить шаблон.
- Insert – вставка строки выражения перед текущей.
- Alt+Insert – вставка строки выражения в конец сетки выражений.
- Delete – удалить текущую строку выражения.

Ниже перечислены стандартные действия по клавишам и комбинациям клавиш в форме ввода формул.

- F1 – показ справки по выделенной функции.
- F3 – поиск сущности в текущем выражении и установка курсора на эту сущность в списке сущностей.
- Ctrl+J – показ меню сниппетов.
- Ctrl+Space – показ списка функций.