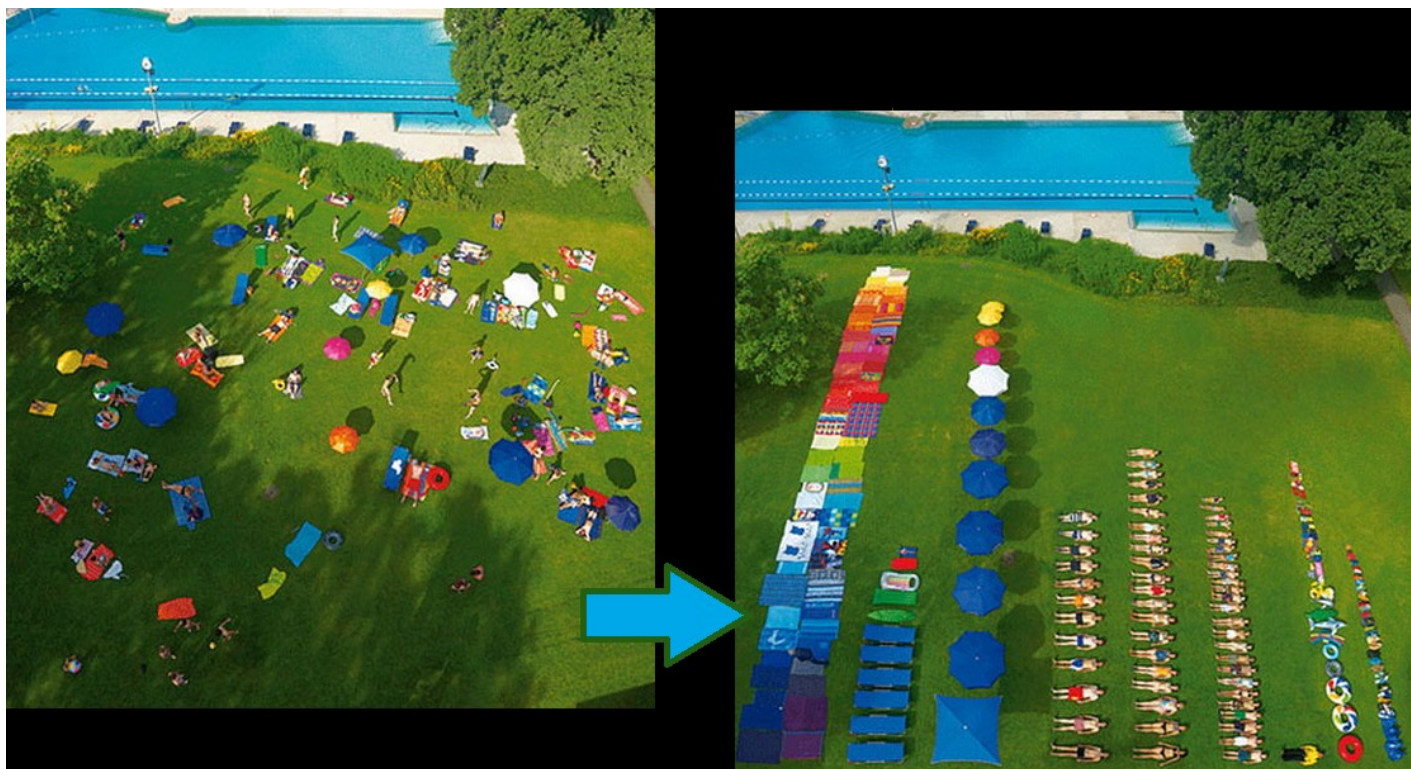


Хранение и повторное использование текстов запросов.



С появлением 1С версии 8.0, язык запросов получил закономерное развитие. Как известно, разработчики платформы ушли с пути “изобретения велосипеда”, и отказались от собственного механизма запросов с уникальным синтаксисом. Теперь язык запросов в 1С является, по сути, русским (ладно, билингвальным) диалектом SQL. А фирма 1С всячески рекомендует использовать запросы практически для любого извлечения данных.

В современных типовых и не совсем типовых конфигурациях, запросы используются повсеместно, причем не редки случаи, когда тексты запросов склеиваются из кусочков во время исполнения кода. Такие “лоскутные” запросы крайне сложно отлаживать, сам текст запроса еще можно перехватить в отладчике, непосредственно перед исполнением, но вот чтобы понять, почему текст запроса получился именно таким, придется изрядно попотеть.

Другая проблема - повторное использование текстов запросов. Может быть так, что один и тот же запрос используется в нескольких разных местах. Хорошим тоном будет вызвать в таких местах функцию, которая вернет текст запроса, но вместо этого, запрос просто копируется, или же вообще переписывается из модуля в модуль. В такой ситуации, если в запрос нужно внести изменения, понадобится долго и трудно искать все случаи использования этого запроса. Мало того, что это скучная, монотонная и механическая работа, так еще и велик шанс что-нибудь упустить.

А можете ли вы, например, сказать, сколько различных, уникальных текстов запросов используется в вашей конфигурации, или хотя бы обработке? А если рассматривать пакетный запрос как несколько запросов, то не возникает ли у вас ситуации, когда два, три, пять пакетных запросов отличаются только последним запросом в пакете?

В связи с вышеизложенным, я задумался об организации текстов запросов и в результате моих размышлений родилась идея, суть которой я попытаюсь изложить в этой статье.

Первое, что пришло мне в голову, это организовать общий модуль, или просто выделить место в модуле объекта, где будут размещаться функции, возвращающие тексты запросов. Но такой подход мне сразу не понравился. Не будем заострять внимание на деталях, - просто интуитивно не понравился и всё! Поразмышляв на эту тему, вы сами сможете найти нужное вам количество недостатков.

В конечном итоге я обратил свой взор на макеты, а именно - на текстовые макеты. Практически все объекты конфигурации (да, не абсолютно все, но очень многие), могут содержать в своих метаданных макеты. Тексты запросов так и просятся, чтобы их размещали в текстовых макетах объекта, если эти запросы имеют отношение исключительно к объекту. Если же запрос носит общий характер для всей конфигурации, имеет смысл сохранить его как один из общих макетов конфигурации.

Формы

ОсновныеДополнительные

Форма обработки: Форма

+

✖

↕

↕

↕

Реквизиты

Табличные части

Формы

Форма

Макеты

ЗапросТаблицаДляГенерацииДокументов

ЗапросТаблицаСпецификацийСырьяИПродукции

ЗапросДатаАктуальностиОстатков

ЗапросСписокДатПер

ЗапросОстаткиСырьяИ

ЗапросОборотыСырьяИ

ЗапросОстаткиНаНач

ЗапросКалендарь_Ме

ЗапросСписокСырьяИ

ЗапросНарастающиеИ

ЗапросОборотыСырьяИ

ЗапросОстаткиИОбор

ЗапросНарастающиеИ

ЗапросОборотыСырьяИ

ЗапросОстаткиИОбор

ЗапросСформировать

ЗапросСформировать

Конструктор макета

Имя:ЗапросВыбокаОборо

Синоним:Запрос выбока оборо

Комментарий:

Выберите тип макета:

☐Табличный документ

☒Текстовый документ

☐Двоичные данные

☐Active document

☐HTML документ

☐Географическая схема

☐Графическая схема

☐Схема компоновки данных

☐Макет оформления компоновки данных

Загрузить из файла:

ГотовоОтменаСправка

Еще один способ, который имеет право на жизнь - создать обработку, выполняющую роль “библиотеки запросов”, и хранить общие запросы в ее макетах. Таких обработок может быть несколько, для лучшей организации и группировки текстов запросов. Также можно комбинировать: общий макет с общими для конфигурации запросами и, отдельно - обработки с запросами, относящимися к определенным подсистемам.

Далее, когда вам нужно выполнить какой-либо запрос, вы просто извлекаете текст нужного запроса конструкцией:

- для случая, когда запрос в макете объекта:

```
мОбъектЗапроса.Текст = ЭтотОбъект.ПолучитьМакет ("ЗапросТаблицаСписокДатПериода").ПолучитьТекст ();
```

- для случая, когда запрос в общем макете конфигурации:

```
мОбъектЗапроса.Текст = ПолучитьОбщийМакет ("ЗапросТаблицаСписокДатПериода").ПолучитьТекст ();
```

- для случая, когда запрос в макете специальной обработки:

```
мОбъектЗапроса.Текст =  
Обработки.ЗапросыПодсистемыСклад.ПолучитьМакет ("ЗапросВыборкаОстаткиНоменклатуры").ПолучитьТекст ();
```

Если вам нужно построить пакетный запрос, вы просто склеиваете его из разных макетов:

```
мОбъектЗапроса.Текст = мОбъектЗапроса.Текст +  
ЭтотОбъект.ПолучитьМакет ("ЗапросВыборкаОстаткиСырьяИГотовойПродукцииНаНачалоПериода").ПолучитьТекст ();
```

Ну, думаю, идея понятна. Хотелось бы немного остановиться на присвоении имен макетам с запросами. Естественно, имена могут быть любыми, и если вы потомок Штирлица, или сотрудник КГБ, вы можете использовать имена типа “31”, “32” - смотрите как интересно, русская обфускация - бессмысленная и беспощадная - в действии! Враг точно будет в шоке. Но, для лучшего восприятия кода, я использую такую схему имени:

1. Слово **“Запрос”** - все макеты, содержащие тексты запросов, начинаются со слова “Запрос”, чтобы можно было отличить их от других макетов.
2. **“Таблица”** или **“Выборка”** - после слова “Запрос” следует слово “Таблица” или “Выборка”. Слово “Таблица” означает, что запрос содержит команду “Поместить” и возвращает временную таблицу, “Выборка” - соответственно означает, что результат запроса нужно обрабатывать как обычно.
3. *ИмяВременнойТаблицы* или *Краткое описание выборки* - Заканчивается имя макета именем временной таблицы, или кратким описанием содержимого выборки.

Таким образом, имя макета приобретает, например, вид: "ЗапросТаблицаСписокДатПериода"; встретив его в коде, или увидев в дереве метаданных, мы поймем, что в этом макете содержится текст запроса, результат выполнения которого будет помещен во временную таблицу с именем "СписокДатПериода".

Открыв этот макет на просмотр, мы увидим:

```
//Тип запроса: Самостоятельный универсальный
//Требования : Передача параметров ДатаНачалаПериода (Тип "Дата") и ДатаОкончанияПериода (Тип "Дата")
//Результат : Временные таблицы "СписокДатПериода" и "Цифры"

ВЫБРАТЬ 0 КАК Цифра ПОМЕСТИТЬ Цифры ОБЪЕДИНИТЬ
ВЫБРАТЬ 1 ОБЪЕДИНИТЬ
ВЫБРАТЬ 2 ОБЪЕДИНИТЬ
ВЫБРАТЬ 3 ОБЪЕДИНИТЬ
ВЫБРАТЬ 4 ОБЪЕДИНИТЬ
ВЫБРАТЬ 5 ОБЪЕДИНИТЬ
ВЫБРАТЬ 6 ОБЪЕДИНИТЬ
ВЫБРАТЬ 7 ОБЪЕДИНИТЬ
ВЫБРАТЬ 8 ОБЪЕДИНИТЬ
ВЫБРАТЬ 9;

////////////////////////////////////
ВЫБРАТЬ
    ДОБАВИТЬКДАТЕ(&ДатаНачалаПериода, ДЕНЬ, СписокДней.Дней) КАК Период
ПОМЕСТИТЬ СписокДатПериода
ИЗ
    (ВЫБРАТЬ
        СотниТысяч.Цифра * 100000 + ДесяткиТысяч.Цифра * 10000 + Тысячи.Цифра * 1000 + Сотни.Цифра * 100 +
        Десятки.Цифра * 10 + Единицы.Цифра КАК Дней
    ИЗ
        Цифры КАК СотниТысяч
        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Цифры КАК ДесяткиТысяч
        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Цифры КАК Тысячи
        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Цифры КАК Сотни
        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Цифры КАК Десятки
        ВНУТРЕННЕЕ СОЕДИНЕНИЕ Цифры КАК Единицы
        ПО (Единицы.Цифра <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ))
        ПО (Десятки.Цифра * 10 <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ))
        ПО (Сотни.Цифра * 100 <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ))
        ПО (Тысячи.Цифра * 1000 <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ))
        ПО (ДесяткиТысяч.Цифра * 10000 <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ))
    ГДЕ
        СотниТысяч.Цифра * 100000 + ДесяткиТысяч.Цифра * 10000 +
        Тысячи.Цифра * 1000 + Сотни.Цифра * 100 + Десятки.Цифра * 10 +
        Единицы.Цифра <= РАЗНОСТЬДАТ(&ДатаНачалаПериода, &ДатаОкончанияПериода, ДЕНЬ)
    ) КАК СписокДней
;
```

Отсутствие вертикальных черточек и подсветка синтаксиса повышают восприятие. Копирование в консоль запросов, для отладки, и обратно - тоже упрощается.

Взглянув на список макетов в дереве метаданных, мы сразу можем получить представление о том, какими запросами оперирует объект. Естественно, чем аккуратнее вы будете подходить к именованию макетов, тем более информативным будет этот список.

Нужный запрос будет легко найти, даже просто взглянув на дерево метаданных. Знаете, бывает такое зудящее чувство: "Где-то я писал подобный запрос...", - как в этом случае у вас продвигались поиски? Я, например, иногда все переписывал заново.

Код в модулях станет более лаконичным. Согласитесь, огромные блоки текста с запросами, прямо в коде, порой просто мешают охватить взглядом логику алгоритма. Убрав запросы с глаз долой, мы делаем код более наглядным.

Да, иногда для понимания кода нужно знать, что выдает запрос на выходе. Эту информацию лучше описать в комментарии, коротко и по-человечески, сразу перед получением текста из макета.

Можно развить идею и использовать в тексте макета шаблоны, подменяя их перед выполнением запроса. Мне самому пока такая мысль не нравится, я считаю, что лучше использовать два разных макета, - раз уж возникла нужда в разных запросах. Но кто мешает вам попробовать?

Возможно, идея ошибочна, возможно она противоречит неким стандартам 1С, я публикую ее здесь, отчасти и для того, чтобы открыть конструктивную дискуссию по этому вопросу. Ни в типовых конфигурациях, ни в авторских разработках я такого подхода пока не встречал.

Автор:

Статья опубликована по адресу:

Профиль автора на сервере Infostart:

Водаков Сергей Вячеславович

<http://infostart.ru/public/142683/>

<http://infostart.ru/profile/16711/>