Qlik Sense. Соглашения о Qlik Script

Table of Contents

1 Цели 6

2 Структура приложения 7

2.1 Секция "Информация" 7

2.2 Секции "Инициализация" и "Main" 7

2.3 Секция "Процедуры" 8

2.4 Секции скрипта 8

2.5 Секция "Конец" 8

2.6 Группировка секций 9

2.7 Секция-разделитель 9

3 Отступы и пропуски 10

3.1 Отступы 10

3.2 Длина строки 10

3.3 Перенос строк 10

3.4 Присвоение переменных 10

3.5 Алиасы (AS) 10

3.6 Пустые строки 11

3.7 Пробелы 11

4 Комментарии 12

4.1 Блочный комментарий 12

4.2 Однострочные комментарии 12

4.3 Комментарии в конце строки 12

4.4 Комментарии в конце строки 13

4.5 Комментирование кода 13

4.6 Комментарии в таблицах INLINE 13

5 Выражения 14

5.1 Простые операторы 14

5.2 Инициализация и отчистка переменных 14

5.3 Выражения в переменных и master items 14

5.4 Оператор LOAD 15

5.4.1 Исключение для GROUP BY и ORDER BY 15

5.4.2 Порядок полей в операторах LOAD 15

5.4.3 INLINE загрузка 16

5.5 Включаемые скрипты 16

5.6 Оператор IF..THEN..ELSEIF..END IF 16

5.7 Оператор SWITCH..CASE..DEFAULT..END SWITCH 16

5.8 Оператор DO..EXIT DO..LOOP 16

5.9 Оператор FOR..NEXT и FOR EACH..NEXT 17

5.10 Операторы LOAD \* и FOR EACH 17

6 Соглашения об именовании 18

6.1 Именование приложений и задач 18

6.2 Именование файлов 18

6.2.1 Расширения файлов 18

6.2.2 Правила именования файлов 19

6.3 Именование таблиц и полей 19

6.4 Именование переменных 20

6.5 Именование процедур 20

7 Хорошие практики написания скриптов 21

8 Безопасность (Section Access) 23

9 Процедуры 24

10 Библиотеки 25

11 Примеры скриптов 26

11.1 Пример секции "Информация" 26

11.2 Пример секции "Инициализация" 26

* [Цели](#scroll-bookmark-3)
* [Структура приложения](#scroll-bookmark-4)
  + [Секция "Информация"](#scroll-bookmark-5)
  + [Секции "Инициализация" и "Main"](#scroll-bookmark-6)
  + [Секция "Процедуры"](#scroll-bookmark-7)
  + [Секции скрипта](#scroll-bookmark-8)
  + [Секция "Конец"](#scroll-bookmark-9)
  + [Группировка секций](#scroll-bookmark-10)
  + [Секция-разделитель](#scroll-bookmark-11)
* [Отступы и пропуски](#scroll-bookmark-12)
  + [Отступы](#scroll-bookmark-13)
  + [Длина строки](#scroll-bookmark-14)
  + [Перенос строк](#scroll-bookmark-15)
  + [Присвоение переменных](#scroll-bookmark-16)
  + [Алиасы (AS)](#scroll-bookmark-17)
  + [Пустые строки](#scroll-bookmark-18)
  + [Пробелы](#scroll-bookmark-19)
* [Комментарии](#scroll-bookmark-20)
  + [Блочный комментарий](#scroll-bookmark-21)
  + [Однострочные комментарии](#scroll-bookmark-22)
  + [Комментарии в конце строки](#scroll-bookmark-23)
  + [Комментарии в конце строки](#scroll-bookmark-24)
  + [Комментирование кода](#scroll-bookmark-25)
  + [Комментарии в таблицах INLINE](#scroll-bookmark-26)
* [Выражения](#scroll-bookmark-27)
  + [Простые операторы](#scroll-bookmark-28)
  + [Инициализация и отчистка переменных](#scroll-bookmark-29)
  + [Выражения в переменных и master items](#scroll-bookmark-30)
  + [Оператор LOAD](#scroll-bookmark-31)
    - [Исключение для GROUP BY и ORDER BY](#scroll-bookmark-32)
    - [Порядок полей в операторах LOAD](#scroll-bookmark-33)
    - [INLINE загрузка](#scroll-bookmark-34)
  + [Включаемые скрипты](#scroll-bookmark-35)
  + [Оператор IF..THEN..ELSEIF..END IF](#scroll-bookmark-36)
  + [Оператор SWITCH..CASE..DEFAULT..END SWITCH](#scroll-bookmark-37)
  + [Оператор DO..EXIT DO..LOOP](#scroll-bookmark-38)
  + [Оператор FOR..NEXT и FOR EACH..NEXT](#scroll-bookmark-39)
  + [Операторы LOAD \* и FOR EACH](#scroll-bookmark-40)
* [Соглашения об именовании](#scroll-bookmark-41)
  + [Именование приложений и задач](#scroll-bookmark-42)
  + [Именование файлов](#scroll-bookmark-43)
    - [Расширения файлов](#scroll-bookmark-44)
    - [Правила именования файлов](#scroll-bookmark-45)
  + [Именование таблиц и полей](#scroll-bookmark-46)
  + [Именование переменных](#scroll-bookmark-47)
  + [Именование процедур](#scroll-bookmark-48)
* [Хорошие практики написания скриптов](#scroll-bookmark-49)
* [Безопасность (Section Access)](#scroll-bookmark-50)
* [Процедуры](#scroll-bookmark-51)
* [Библиотеки](#scroll-bookmark-52)
* [Примеры скриптов](#scroll-bookmark-53)
  + [Пример секции "Информация"](#scroll-bookmark-54)
  + [Пример секции "Инициализация"](#scroll-bookmark-55)

# Цели

* определить внутренние требования к оформления кода на Qlik Script

# Структура приложения

Хорошей практикой является использование секций (вкладок) в скрипте для логического и визуального разделения различных частей, упрощая просмотр скрипта в ходе разработки и поддержки. В зависимости от назначения и сложности приложения может быть применена своя логика, общие части скрипта приведены ниже:

* Безопасность (Section Access)
* Календарь с датам и иными атрибутами
* Секция для инициализации
* Секции для источников данных
* Секции для измерений
* Секции для ключевых показателей
* Секции для дополнительных таблиц
* Секция для отчистки и завершения

Предлагаемая схема описана в подразделах ниже.

## Секция "Информация"

Каждый скрипт должен содержать первую вкладку "Информация" с блоком комментариев, который включающим общую информацию о приложении и историю изменений, см. пример в разделе "Примеры скриптов"

Допустимые версии имени секции в латинице - **Information**, **Info**.

## Секции "Инициализация" и "Main"

Секция инициализации используется для определения следующих необязательные параметров:

1. Глобальные настройки среды - например, расположение папок
2. Внешние скрипты - библиотеки кода и т.д.
3. Константы. Сценарий Qlik Sense не имеет реального определения константы, поэтому используются переменные с префиксом const (подробнее об этом в разделе "Переменные")
4. Переменные - глобальные переменные, используемые внутри скрипта, или, при необходимости, переопределяющие переменные, которые были определены во внешних скриптах (например, чтобы переопределить конкретный параметр среды)

Секция "**Main**", создаваемая автоматически при создании приложения может быть объединена со скриптом инициализации переменных и прочих значений, необходимых для приложения, при этом секцию можно переименовать в "**Инициализация**" или оставить как есть, разместив на втором месте после вкладки "Информация". В ином случае, секцию "Инициализация" стоит разместить после "Main".

Допустимые версии имени секции в латинице - **Initialize**, **Init**.

## Секция "Процедуры"

Процедуры или подпрограммы (**Sub**routine), которые используются в нескольких приложениях Qlik Sense, должны быть помещены во внешний файл и включены на вкладке "Инициализация". Любые подпрограммы, которые являются локальными для приложение Qlik Sense (т.е. используются в этом и только в этом приложении) следует поместить в секцию с названием "Процедуры". Эта секцию является размещается после секции "Инициализация". Если имеется много локальных подпрограмм, секция "Процедуры" может быть продублирована для каждой отдельной процедуры или группы процедур, что визуально рекомендуется сгруппировать с помощью вкладок группировки (обсуждается ниже), а каждому экземпляру вкладки необходимо дать ёмкое название, соответствующее логике.

Допустимые версии имени секции в латинице - **Subs**, **Sub**.

## Секции скрипта

Секции предоставляют удобный механизм для организации содержимого скрипта:

* Каждая секция должны быть посвящена одной из основных функциональных назначений (например, загрузка таблицы, выполнение вычислений, выполнение нескольких связанных расчетов)
* В идеале, стоит стремиться к 80-100 строк кода на одну секцию в качестве общего практического правила. Ограничение вкладок этим размером улучшит организацию кода.
* Каждая вкладка должна иметь уникальное имя (уникальное внутри приложения). Это особенно полезно при экспорте скрипта в файл, так как в этом процессе все секции объединяются
* Имя секции должно быть ёмким и описательным, предпочтительно в форме подлежащего глагола. Например:
* Извлечь SAP
* Сумма продаж
* Экспорт таблиц
* Кроме того, комментарий к заголовку должен быть размещен в верхней части вкладки (за исключением вкладок по умолчанию). См. пример в разделе "Комментарии"

## Секция "Конец"

Каждый сценарий должен содержать вкладку "Конец". Эта вкладка используется для обозначения конца скрипта и поэтому обычно последняя вкладка в скрипте.

Вкладка "Конец" должна содержать:

1. отчистку переменных скрипта:
   1. **vVariable=;**
2. оператор выхода
   1. **EXIT SCRIPT;**

Допустимые версии имени секции в латинице - **Exit**, **End**.

## Группировка секций

Если несколько секций относятся к одной и той же предметной области или выполняют аналогичные операции, это можно добавить группирующие секции вокруг них. Это могут быть просто пустые секции - стартовая секция с именем имя группы, за которым следуют два знака "больше" (>>), завершающая секция с именем группы и двумя знаками "меньше" (<<). Например:

* Extract >>
* Extract 1
* Extract 2
* << Extract

## Секция-разделитель

Добавив секцию скрипта с названием, например, "- - - - -" можно визуально разделить секции на группы

# Отступы и пропуски

## Отступы

Используйте отступы, чтобы сделать скрипт более удобочитаемым разработчиками.

Конструкция отступов (пробелы или знаки табуляции) - выбор разработчика. Пробелы лучше в плане переносимости в другие редакторы, табуляция требует меньшего количества нажатий клавиш.

Рекомендуемая единица отступа - **4 пробела**.

## Длина строки

Рекомендуемая длина строки - **80 символов**. При разработке на бОльшем и/или широкоэкранном мониторе, стоит проверить просмотр скрипта на экранах или в окне меньшего размера, чтобы убедиться в читаемости кода.

## Перенос строк

Если выражение не помещается на одной строке, его нужно разбить в соответствии со следующими общими принципами:

* Разрыв после запятой
* Разрыв перед оператором (+, - и т.д.)
* Выровнять новую строку с началом выражения на уровне предыдущей строки
* При наличии скобок, нужно стремиться использовать разрывы для скобок более высокого уровня, объединяя более низкие уровни в отдельные строки
* Если приведенные выше правила приводят к запутанному коду или коду, сжатому до правого поля, вместо этого просто сделайте отступ 4 пробела.

## Присвоение переменных

Рекомендуется **не добавлять** пробелы и/или знаки табуляции для выравнивания знака присваивания при объявлении переменных.

## Алиасы (AS)

Рекомендуется **выравнивать** алиасы (псевдонимы). Это вызывает дополнительные накладные расходы во время разработки, но созданное приближенное к табличному представление, делает сценарий более читабельным. Стоит иметь в виду, что есть риск пропустить поля без псевдонимов, особенно если они предшествуют или следуют за многострочным выражением. Этого можно избежать, если создать псевдоним с таким же именем (Facility AS Facility).

|  |
| --- |
| // Load trouble ticket details from QVD  TroubleTicketDetail:  LOAD  Ticket                 AS %TicketKey,  Ticket                 AS FaultID,  [Event Type]           AS EventType,  date(Date, 'MM/DD/YYYY') AS ActualDate,  Facility               AS Facility,  Client AS Contract,  Type AS Type,  Duration AS [Duration (min)],  1 AS TroubleTicketDetailCounter  FROM  [lib://Folder/File.qvd] (qvd)  WHERE  Date >= '12/01/2022'  AND Type = 'Incident' // logic from QOS report  AND Duration > 0  ; |

## Пустые строки

Пустые строки улучшают читаемость, выделяя разделы кода, которые логически связаны. Две пустые строки следует всегда использовать в следующих случаях:

* Между секциями файла скрипта
* После оператора **LOAD**, если только следующий оператор **LOAD** не использует оператор **CONCATENATE** или **JOIN** префикс, в этом случае используется одна строка

Одна пустая строка всегда должна использоваться в следующих случаях:

* Перед блочным или однострочным комментарием (см. раздел "Комментарии" выше)
* Между логическими разделами внутри скрипта для улучшения читабельности

## Пробелы

Пробелы следует использовать в следующих случаях:

* После запятых в списках аргументов должен стоять пробел (см. раздел ниже)
* До и после оператора присваивания должен стоять пробел, т.е. знак равенства **"="**
* Все бинарные операторы должны быть отделены от своих операндов пробелами. Пробелы должны никогда не отделяйте унарные операторы, такие как унарный минус, от их операндов.

# Комментарии

Комментарии являются важной частью разработки, позволяющей дать обзор функциональности и реализованной логики в краткой, но ясной форме.

Рекомендуется формулировать комментарии как полные предложения, начинающееся с большой буквы, за исключением случаем коротких заметок, комментариев в конце строки и т.п. Нужно стараться явное указывать то, что необходимо получить в результате работы алгоритмов, это значительно повысит читабельность скрипта.

В ключевых местах изменений в коде скрипта необходимо добавлять комментарий с датой изменения.

## Блочный комментарий

Комментарии следует добавлять в начале каждой секции с кратким описания назначения функциональности:

|  |
| --- |
| //------------------------------------------------------------------------------  // Описание функциональности  //------------------------------------------------------------------------------    /\*  Описание функциональности  \*/ |

## Однострочные комментарии

Короткие комментарии могут располагаться на одной строке с отступом, соответствующим следующей строке скрипта. Если комментарий не может быть записан в одну строку, он должен соответствовать формату блочного комментария (см. предыдущий раздел).  
Однострочный комментарий использует две косые черты (//) и должен предваряться пустой строкой.

|  |
| --- |
| IF vScript.ExtractData = 1 THEN  // Handle the condition.  LOAD \* FROM file.csv;  END IF |

## Комментарии в конце строки

Очень короткие комментарии могут появляться в одной строке с кодом, который они описывают, но их следует сдвинуть, так, чтобы отделить их от выражений. Если в фрагменте кода появляется более одного короткого комментария, все они должны иметь один и тот же отступ. Однако, не стоит сильно увлекаться "ассемблерным" стилем комментирования каждой строки скрипта завершающим комментарием.

|  |
| --- |
| IF vCondition = 0 THEN  LOAD \* FROM file.csv; // Handle the condition.  ELSE  LOAD \* FROM file2.csv; // This file must already exist.  END IF |

## Комментарии в конце строки

Однострочный комментарий действует до конца строки. Им можно закомментировать полную строку или часть строки. Его не следует использовать для блочных комментариев, однако его можно использовать в нескольких последовательных строках для комментирования части кода.

|  |
| --- |
| // Remove the following comments before production use  //MyTable:  //LOAD \* INLINE [  // X, Y  // A, 1  // B, 2  //]; |

## Комментирование кода

Комментирование кода (его части) - это очень эффективный метод при тестировании и отладке. Однако в продакшене стоит избегать закомментированного кода Qlik Script, он должен на 100% "чистым" для запуска, и любые неясности в "подозрительном" коде должны решаться в процессе разработки и обеспечения качества. Закомментированный код в производственном скрипте не является хорошей практикой проектирования и разработки.

## Комментарии в таблицах INLINE

Особое внимание следует уделить **INLINE** таблицам и комментариям. Комментарии не могут быть помещенным между квадратными скобками "[…]" определения **INLINE** таблицы. Любой текст между этими скобками будут интерпретироваться как данные.

MyTable:  
LOAD \* INLINE [  
X, Y  
A, 1 // Assign value 1 to item A.  
B, 2  
C, 3  
];

# Выражения

Регистр ключевых слов должен соответствовать вариантам, предлагаемым редактором по умолчанию.

## Простые операторы

Простые операторы могут и должны занимать в сценарии только одну строку. Типичными примерами простых утверждений являются:

* Выражения с одним ключевым словом
* Инициализация переменных
* Простые операторы загрузки или выбора, которые не включают списки полей или очень короткие списки полей

## Инициализация и отчистка переменных

В общем случае, глобальные переменных должны быть объявлены в [секции "Инициализация"](#scroll-bookmark-56), а переменные, использованные только в рамках скрипта (скриптовые переменных) должны быть отчищены в ходе выполнения скрипта и/или в [секции "Конец"](#scroll-bookmark-57).

## Выражения в переменных и master items

Использование выражений в Qlik Sense может быть интенсивным, особенно при наличии сложного пользовательского интерфейса. Растет потребность в более эффективной обработке этих выражений, и это может быть достигнуто за счет использования переменных совместно с master items.

Причины хранения выражений в переменных:

* Чтобы добиться повторного использования: формула для такого показателя, как "Продажи", обычно остается одинаковой во всем документе Qlik Sense, поэтому нет смысла писать ее на каждой диаграмме
* Обеспечить согласованность формул: избегая риска использования разных формул для расчета одного и того же показателя
* Чтобы обеспечить единую точку для применения изменений: если и когда необходимо изменить формулу, достаточно изменить только одну переменную или master item, и все диаграммы и другие объекты, которые ссылаются на эту переменную, последуют за ней
* Чтобы позволить конечному пользователю вносить изменения через поле ввода, когда это необходимо. Это могут быть цели для ключевых показателей эффективности или общих параметров

В зависимости от логики, может потребовать использование как **Let**-, так и **Set**-переменных. Расширение со знаком **$** указывает, что строка, содержащаяся в переменной, является формулой, которую необходимо вычислить.

Существует два простых метода изменения значения этой переменной из среды в среду и из приложения в приложение:

1. Разработчик и/или администратор вручную изменяет значение переменных в скрипте
2. Использовать текстовые файлы **Include** с выражениями **Let/Set** или таблицу (Excel или БД) с соответствующими выражениями в них. В этом случае каждая среда должны иметь свою собственную версию выражений с переменными, а в приложение загружаются выражения, соответствующее среде. В результате, логика выражений храниться в Excel, плоском файле или в базе данных, что позволяет централизованно управлять логикой показателей.

*NB. при решении в пользу БД необходимо обеспечить инструментарий для ввода и управления данным*

## Оператор LOAD

Каждый скрипт включает один или несколько операторов **LOAD**. Они представляют собой механизм загрузки данных в ассоциативную модель Qlik. Оператор **LOAD** обычно содержит ряд ключевых слов, таких как: **LOAD**, **SQL**, **SELECT**, **FROM**, **WHERE**, **GROUP BY** и т. д.

Эти ключевые слова должны появляться отдельно **на отдельной строке**, а каждая последующая строка должна иметь отступ в **4 пробела**. Кроме того, каждое поле в списке **LOAD**/**SELECT** должно находиться на отдельной строке. Аналогично для каждого условия в предложении **WHERE**. Закрывающая точка с запятой всегда ставится на отдельной строке в самом начале. Это упрощает добавление и удаление предложений **WHERE**, **GROUP BY**, **ORDER BY** и вводит естественные пробелы, облегчающие чтение сценария. Ставить запятую в конце поля или перед ним - выбор разработчика, важнее единообразие и выбор в пользу упрощения поддержки.

### Исключение для GROUP BY и ORDER BY

Исключение из правила "одно поле на строку" можно сделать для простых аргументов **GROUP BY** и **ORDER BY**. Если есть несколько простых аргументов, их можно поместить в одну строку после ключевого слова.

### Порядок полей в операторах LOAD

Рекомендуется придерживаться следующего порядка, в котором поля появляются в операторе **LOAD**. Особенно в больших таблицах это упрощает поиск соответствующих полей:

* **Первичный ключ**: ключ, однозначно идентифицирующий одну строку в таблице. В идеале это должно быть одно поле, но оно может состоять из нескольких полей в случае таблиц фактов
* **Внешние ключи**: ключи, которые используются для связывания таблицы с другой таблицей. Обычно это первичный ключ таблицы, на которую делается ссылка
* **Атрибуты**: описательные метки/даты/классификации/и т. д. Связанные (по смыслу, назначению) поля должны быть сгруппированы вместе
* **Показатели**: поля, содержащие (полу-)аддитивные факты
* **Поля флагов**: поля-индикаторы, которые указывают, соответствует ли строка определенному условию (значение 1) или нет (значение 0). Также см. раздел "Именование переменных и полей"
* **Поля-счетчики**: поле, содержащее значение 1 для каждой строки, которое используется для простого подсчета количества строк в выделенном фрагменте и 0 или NULL для остальных записей
* **Поля метаданных**: поля, содержащие информацию о данных. Например, поля "Действительно с" и "Действительно до", используемые для описания периода, в течение которого запись действительна в сценарии SCD типа 2. Или FactType, чтобы указать, из какой таблицы фактов исходит строка в сценарии составной таблицы фактов.

### INLINE загрузка

Это один из альтернативных способов загрузки данных в ассоциативную модель, рассчитанный на небольшой объем данных, предназначенных, например, для сортировки или мэппинга значений, оптимизации загрузки по **EXISTS()**. Inline-таблице необходимо дать название и включить имена полей в первой строке внутри квадратных скобок "[..]".  Закрытие точка с запятой может стоять на той же строке, что и квадратная закрывающая скобка.

## Включаемые скрипты

Существeет два вариант оператора включения:

* **$(include=script.qvs);** - во время выполнения **include** Qlik не проверяет, действительно ли существует включенный файл. Если файл не существует, скрипт продолжит работу без каких-либо ошибок
* **$(must\_include=script.qvs);** - если включаемый файл является обязательным, следует использовать **must\_include**, который выполняет проверку во время выполнения и выдает ошибку, если файл не существует.

## Оператор IF..THEN..ELSEIF..END IF

Каждое из ключевых слов: **IF**, **THEN**, **ELSEIF**, **ELSE** и **END IF** должно начинаться с отдельной строке с одинаковым отступом. Сразу после ключевого слова **IF** или **ELSEIF** поместите условие и завершите строку ключевым словом **THEN**.

## Оператор SWITCH..CASE..DEFAULT..END SWITCH

Каждое из ключевых слов: **SWITCH** и **END SWITCH** должно начинаться с отдельной строки с одинаковым отступом. Ключевые слова **CASE** располагаются на отдельной строке с отступом в **четыре пробела** правее уровня открывающего **SWITCH**. Аналогично для ключевого слова **DEFAULT**.

## Оператор DO..EXIT DO..LOOP

Каждое из ключевых слов: **DO** и **LOOP** должно начинаться с отдельной строки с одинаковым отступом. Ключевые слова **WHILE** или **UNLESS** и условие следует указывать в той же строке, что и **DO**. Если включено условие прерывания (ключевое слово **EXIT DO**), его следует поместить на отдельной строке. Он должен иметь отступ, как и любой другой оператор внутри цикла.

## Оператор FOR..NEXT и FOR EACH..NEXT

Каждое из ключевых слов: **FOR**, **FOR EACH** и **NEXT** должно начинаться с отдельной строки. Оператор счетчика следует размещать в той же строке, что и **FOR**. Если требуется указать размер шага приращения счетчика, включите ключевое слово **STEP** в ту же строку. Если включено условие прерывания (ключевое слово **EXIT FOR**), его следует поместить на отдельной строке. Он должен иметь отступ, как и любой другой оператор внутри цикла.

## Операторы LOAD \* и FOR EACH

В операторе **LOAD** можно использовать подстановочный символ **"\*"** для пакетной загрузки нескольких файлов. Например, следующий оператор загрузит поля CustomerID и SalesAmount из каждого CSV-файла в папке данных.

|  |
| --- |
| Data:  LOAD  CustomerID,  SalesAmount  FROM  [data\\*.csv]  ; |

Это очень полезная возможность, но у нее есть свои ограничения, когда речь идет о гибкости обработки файлов, например, в этом случает нельзя использовать предыдущую загрузку. В свою очередь, оператор **FOR EACH ... NEXT** предоставляет альтернативную возможность, обеспечивающую б***о***льшую гибкость:

|  |
| --- |
| FOR EACH vFilename IN fileList('data\\*.csv')  Data:  LOAD      CustomerID,      SalesAmount  FROM      [$(vFilename)]  ;  NEXT |

# Соглашения об именовании

## Именование приложений и задач

Приложения и задачи c загрузчиками данных должны быть названы по шаблону "**Extract\_[*ИСТОЧНИК*]\_[*ВИТРИНА*]**", где:

* **Extract\_** - постоянный префикс
* [***ИСТОЧНИК***] - наименование системы-источника
* **[*ВИТРИНА*]** - название витрины, таблицы БД
  + если загрузчик загружает много таблиц, то должно быть указано обобщающее название (Например: Справочники SAP BW)

Приложения и задачи с расчётами (трансформациями) данных должны быть названы начиная с "**Transform\_L*1..N*\_[*НАЗНАЧЕНИЕ*]**", где:

* **Transform\_** - постоянный префикс
* **L*1..N*** - номера уровня трансформации, где ***1..N*** натуральное число, начиная с 1, соответствующее этапу/уровню трансформации
* **[*НАЗНАЧЕНИЕ*]** - название проекта, приложения, являющегося целевым назначением данных

Префиксы и суффиксы могут быть добавлены к рабочим версиям приложений, утверждённые версии приложений в общем случае не должны именоваться с применением суффиксов.

В отношение задач, префиксы и суффиксы могут быть использованы без ограничений при необходимости.

|  |  |
| --- | --- |
| Сокращения | Описание |
| All | Все среды |
| DEV | Среда разработки |
| PRD | Продуктивная среда |
| TST | Тестовая среда |
| MVP | Временная версия (Minimal Viable Product - минимально жизнеспособный продукт) |

## Именование файлов

### Расширения файлов

В общем случае, расширения файлов не должны противоречить стандартным, например:

* .qvf - файл с приложением Qlik Sense
* .qvd - файл данных в формате QVD
* .qvx - файл данных в формате XML
* .qvs - файл с Qlik Script
* .xls, .xlsx - файл MS Excel
* .csv - файл в формате CSV
* .txt - текстовый файл
* .json - файл в формате JSON
* .xml- файл в формате XML

### Правила именования файлов

Основные правила именования файлов:

* Все пользовательские приложения должны быть названы описательными и ёмкими бизнес-терминами с использованием нескольких слов, отображаемыми в веб-хабе Qlik Sense.
* Именование QVD-файлов
* Включаемые Include-файлы должны быть названы согласно практике CamelCase и иметь расширение **.qvs**.

Несколько примеров:

* Дашборд Менеджера.qvf - бизнес-приложение, без префикса
* ДМ Аналитика по погрузке.qvf - бизнес-приложение, без префикса
* Extract\_EPRVCM009.qvf - приложение-загрузчик данных
* Transform\_Аренда.qvf - приложение-трансформатор данных
* EPRVCM009\_202201.qvd
* АрендаРемонты\_202201.qvd
* Goals\_2022.xlsx
* CreateDataProfiling.qvs

## Именование таблиц и полей

Если для таблицы не указано имя, Qlik будет использовать имя исходной таблицы. Иногда это приемлемо, однако, предпочтительнее явно назначать имя каждой загружаемой таблице. Использование соглашения об именовании таблиц в рамках приложения/проекта, обеспечит согласование имен во всех приложениях.

Имена полей с бизнес-данными (используемыми для визуализации) необходимо писать в бизнес-терминологии с пробелами в удобном для пользователя виде, не следует использовать знаки подчеркивания или косые черты.

Следует избегать использования **LOAD \*** или **SELECT** \*, при которых загружаются все поле исходной таблицы. В частности, при загрузке из внешних файлов или баз данных. Изменения в базовой таблице могут привести к тому, что будет возвращено больше полей, чем предполагалось изначально. Исключениями, где **LOAD \*** может быть допустимым, являются резидентные загрузки (когда таблица создается в скрипте, и разработчик может управлять ее содержимым), а также предшествующие загрузки.

Переименование полей необходимо выполнять посредством ключевого слова **AS** в операторах **LOAD**/**SELECT** или **RENAME FIELD(S)**., при наличии таблиц соответствия старых и новых имён полей.

|  |  |  |
| --- | --- | --- |
| Назначение | Правила | Примеры |
| Ключевые поля | начинается с "%" | %CustomerKey |
| Поле-флаг | начинается с "?" | ?YTDFlag |
| Поле-счётчик | начинается с "#" | #YTDFlag |
| Разделитель ключевого поля | разделитель "|" | Company & '|' & Nbr as Key |
| Временные поля и таблицы | начинается с "tmp" | tmpDailyTrans |
| Поля и таблицы сортировки | начинается с "sort" | sortRegion |
| Поля и таблицы мэппинга | начинается с "map" | mapFiledName |

## Именование переменных

Лучше избегать переменных с именем в один символ, кроме случаев переменных "на выброс". Распространённая практика для таких переменных: i, j, k, m, n - для целых чисел; x, y, z - для чисел с плавающей точкой; c, d, e - для строк. Такие переменные не требую префиксов и суффиксов.

Принимаются следующие правила именования переменных:

|  |  |  |
| --- | --- | --- |
| Назначение | Правила | Примеры |
| Переменные скрипта | начинается с "s" | sFieldIndex |
| Переменные для интерфейса | начинается с "v" | vCurrentYear |
| Объединение имён | разделитель "\_" | vFolder\_Transform\_L1 |

## Именование процедур

Процедуры должны быть названы в форме подлежащего глагола согласно практике CamelCase, например:

* CreateCalendar
* StoreQVDdropTables
* CreateLinkTable

# Хорошие практики написания скриптов

* В части оптимизации и построения модели данных см. рекомендации в [Соглашении о моделировании](https://conf.pgk.ru/pages/viewpage.action?pageId=26556031#id-ОмоделяхданныхприложенийQlikSense-Оптимизация)
* Используйте комментарии с меткой **ToDo** в местах, требующих особого внимания и/или пересмотра и дополнения позже, чтобы быстро найти их через поиск
* Никогда не используйте **LOAD \*** в операторе загрузки. Вместо этого перечислите столбцы для загрузки явно, чтобы было очевидно, какие поля будут загружены, и это не изменится при добавлении или удалении новых столбцов из исходных таблиц. Это также помогает разработчикам идентифицировать загруженные поля в скрипте
* **Предыдущие загрузки** (**preceding load**) - использование предыдущих операторов загрузки может упростить скрипт и сделать его более понятным
* При разработке приложений, требующих довольно длительных расчётов, рекомендуется применять двоичную (**BINARY**) загрузку, а затем расширять скрипт. Позже скрипты приложений можно объединить, как разработка будет близка к завершению. Функционально это ничего не меняет, но экономит время при разработке
* При разработке загрузчиков и алгоритмов преобразования данных, конечной целью которых является сохранение данных в QVD рекомендуется отключать индексацию полей посредством **SET CreateSearchIndexOnReload=0;** для сокращения времени выполнения скрипта, т.к. при включенной индексации может затрачиваться довольно значительно время на индексацию
* Используйте **HidePrefix=**; чтобы позволить корпоративному разработчику скрыть ключевые поля и другие поля, которые редко используются дизайнером
* При использовании функции **Applymap()** заполните значение по умолчанию, например "Неизвестно" или "NULL", чтобы пользователи знали, какое значение неизвестно, и могли заполнить его в исходной системе без участия администраторов
* Используйте **Qualify \*** с осторожностью в случае необходимости, чтобы избежать неверного построения модели данных
* Используйте переменные для имени пути вместо использования строковых констант по всему скрипту. Это сокращает обслуживание, а также обеспечивает простой способ поиска путей (при условии, что вы поместите их в секцию "Инициализация", чтобы их было легко найти)
* Включаемые файлы используются для:
  + Повторно используемая логика/подпрограммы
  + Параметры конфигурации (например, языковой стандарт или параметры среды)
  + Строки подключения к источнику данных
  + Общие/глобальные переменные
  + Глобально используемые расчеты (сохраненные как переменные)
* Qlik автоматически объединяет последовательно загружаемые таблицы, если они имеют одинаковое количество полей с одинаковыми именами. Иногда это может быть "удобно", но лучше явно использовать ключевое слово **CONCATENATE**. Это облегчит понимание логики скрипта, особенно когда конкатенация охватывает несколько вкладок.
* При использовании ключевых слов **CONCATENATE**, **JOIN** или **KEEP**, если таблица не указана, Qlik автоматически предполагает, что целевой таблицей является последняя таблица, загруженная перед ключевым словом. Как и в случае с неявной конкатенацией, это снова можно рассматривать как "удобную" возможность, но это также и риск. Если во время последующих обновлений скрипта дополнительный оператор **LOAD** будет размещен непосредственно над ключевым словом **CONCATENATE**, **JOIN** или **KEEP**, Qlik выполнит операцию с этой таблицей вместо исходной, предполагаемой целевой таблицы
* При использовании **JOIN** и **KEEP** необходимо обязательно указывать ключевые поля, т.к. Qlik использует "естественное соединение" при объединении двух таблиц операторами **JOIN** или **KEEP**

|  |
| --- |
| **Естественное соединение** (**natural join**) - это тип соединения, при котором поля, в которых соединяются две таблицы, не определены явно. Любое поле, встречающееся в обеих таблицах, будет использоваться для объединения двух таблиц. Результирующая таблица содержит только один столбец для каждой пары столбцов с одинаковыми именами |

Более поздние изменения скрипта могут привести к тому, что инструкция **JOIN** или **KEEP** будет учитывать больше столбцов, чем предполагалось изначально. Эти ошибки может быть трудно обнаружить. Следует всегда помещать предполагаемые ключи в начало оператора **LOAD** и добавлять комментарий в конце строки, чтобы указать, что это поля, которые должны быть объединены

* Рекомендуется не пренебрегать использованием круглых скобок в выражениях, содержащих смешанные операторы, чтобы избежать проблем с приоритетом операций. Даже если приоритет операций кажется вам понятным, для других он может быть неочевиден - не следует полагать, что другие разработчики знают приоритет так же хорошо, как и вы :-)  
  Приведём наглядный пример:  
  Если переменная содержит выражение, рекомендуется заключить его в круглые скобки. В противном случае, когда в выражении используется несколько переменных, приоритет операций может привести к неожиданному поведению вычислений:

|  |
| --- |
| SET vNetSalesAmount = Sum(TotalAmount) - Sum(ShippingCost);  SET vTaxrate = 0,21; |

Если эти две переменные впоследствии используются в выражении для расчета налога с продаж:

|  |
| --- |
| $(vNetSalesAmount) \* $(vTaxRate) |

Соблюдение приоритета выполнения операций приведет к тому, что выражение будет возвращать неверный результат, т.к. стоимость доставки умножается на налоговую ставку и полученное число вычитается из общей суммы продаж. Заключение выражения в переменной в круглые скобки предотвращает возникновение этой проблемы:

|  |
| --- |
| SET eNetSalesAmount = (sum(TotalAmount) - sum(ShippingCosts)); |

# Безопасность (Section Access)

Рекомендации по использованию Section Access:

* В Section Access всегда используйте функцию **UPPER()** при использовании оператора загрузки, используйте ее для каждого столбца, несмотря ни на что (даже при чтении из .qvd)
* Использовать доменные группы для обеспечения безопасности
* Выносить логику Section Access во включаемые файлы или БД
* Добавьте учетную службы Publisher/Scheduler в таблицу доступа к разделу
* Стремиться к использованию схемы "звезда" для модели данных без таблиц связей. Лучше всего иметь одну таблицу фактов со всеми измерениями, напрямую связанными с фактом. В исключительных случаях следует использовать дополнительные "снежинки".
* В среднем в таблицах фактов должно быть порядка 30–40 столбцов с некоторым отклонение
  + Часто большое количество столбцов возникает при использовании "ролевых атрибутов" (например, несколько дат - отгрузки, приёмки и т.п., связанных с одним фактом), что полезно с точки зрения функциональности, однако, слишком большое количество таких показателей приводит к снижению производительности сервера.

# Процедуры

В процедуры (**Sub**) рекомендуется выносить общую функциональность, используемую в рамках приложения и/или для выстраивания компактного алгоритмы, построенного на вызовах процедур в рамках отдельной секции скрипта.

Параметры процедур должны иметь префикс "**p**", чтобы указать, что они являются локальными для подпрограммы.

В конце каждой процедуры все переменные должны быть обнулены через **vVariable=;**.

Имена локальных переменных должны начинаться с имени процедуры и знака "\_", например, **sTrace\_Space** для **Sub Trace**.

# Библиотеки

Процедуры, общие для более чем одного приложения рекомендуется вынести в библиотеку (одну или несколько, в зависимости от потребностей), при этому, каждую процедуру лучше разместить в отдельном qvs-файле, а также добавить один qvs-файл, включающий (**INCLUDE**) все процедуры библиотеки.

# Примеры скриптов

## Пример секции "Информация"

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Приложение: Дашбор Менеджера КБ  Версия: 1.0  Автор: Иванов И.И.  Дата создания: 2022-01-20  Дата изменения: 2022-04-30  Описание:  Приложение предназначено для ежедневного контроля показателей менеджера КБ за  прошлый и текущий годы. Источником данных является SAP BW и таблица с целевыми  значениями показателей.  --------------------------------------------------------------------------------  Версия: 1.0  Автор: Сидоров С.С.  Дата изменения: 2022-04-30  Изменения:  - GDM-301: добавлены целевые показатели  - GDM-300: исправлены замечания  - Приложение одобрено заказчиком  --------------------------------------------------------------------------------  Версия: 1.0  Автор: Иванов И.И.  Дата изменения: 2021-12-10  Изменения:  - Исходная версия  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

## Пример секции "Инициализация"

|  |
| --- |
| //------------------------------------------------------------------------------  // Переменные и настройки  //------------------------------------------------------------------------------    SET ThousandSep=',';  SET DecimalSep='.';  SET MoneyThousandSep=',';  SET MoneyDecimalSep='.';  SET MoneyFormat='$#,##0.00;($#,##0.00)';  SET TimeFormat='h:mm:ss TT';  SET DateFormat='M/D/YYYY';  SET TimestampFormat='M/D/YYYY h:mm:ss[.fff] TT';  SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';  SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[ Глобальные настройки приложения ]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  $(must\_include=lib://FolderConnection/Folder/File.qvs);    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[ Внешние скрипты ]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  $(must\_include=$(vFolder.Include.Scripts)createcalendarfromfield.qvs);  $(must\_include=$(vFolder.Include.Scripts)createcalendar.qvs);    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[ Константы ]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  SET constYes = Dual('Yes', 1);  SET constNo = Dual('No', 0);    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[ Переменные ]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  SET vScript.ExtractData = 'Y';  SET vScript.TransformData = 'Y'; |