О моделях данных приложений Qlik Sense

Table of Contents

1 Схемы данных 4

2 Островные таблицы (Data Islands) 5

3 Большие модели данных 6

3.1 Все детали в таблице фактов 6

3.2 Цепочки приложение (chaining) 6

3.3 ODAG 7

3.4 Агрегаты и детали в одном приложении 7

4 Оптимизация 8

* [Схемы данных](#scroll-bookmark-3)
* [Островные таблицы (Data Islands)](#scroll-bookmark-4)
* [Большие модели данных](#scroll-bookmark-5)
  + [Все детали в таблице фактов](#scroll-bookmark-6)
  + [Цепочки приложение (chaining)](#scroll-bookmark-7)
  + [ODAG](#scroll-bookmark-8)
  + [Агрегаты и детали в одном приложении](#scroll-bookmark-9)
* [Оптимизация](#scroll-bookmark-10)

# Схемы данных

Основными схемами для построения модели данных в Qlik Sense являются "звезда" (star) и "снежинка" (snowflake).

Таблицы в модели данных условно делятся на таблицы измерения (dimension tables) и таблицы фактов (fact tables).

В общем случае, таблицы измерений будут содержать первичный ключ, атрибуты, поля с флагами и счетчиками. Таблицы фактов будут содержать внешние ключи, меры и (необязательно) поля с флагами и счетчиками. Поля метаданных являются необязательными для обоих типов таблиц.

|  |
| --- |
| **Таблицы измерений** - это таблицы, содержащие описания и атрибуты, предоставляющие контекст показателей (мер), хранящимся в таблице фактов. Например, таблица измерения "Клиент", которая содержит атрибуты для каждого клиента (имя, номер счета, адрес, контактное лицо и т. д.). **Таблицы фактов** содержат показатели. Например, таблица фактов о продажах, содержащая количество проданных продуктов и их цену. Данные в таблицы фактов связываются с данными таблиц измерений с помощью ключевого поля, например %CustomerKey. **Факты** и **измерения** используются в контексте **Dimensional Modeling**, описанным в книге The Data Warehouse Toolkit Ральфа Кимбалла и Марджи Росс (Wiley, ISBN: 978-1118530801). |

Ключевые факторы, влияющие на модель:

* Количество уникальных значений в столбцах
* Количество уникальных значений в ключевых полях
* Количество связей между полями, использованными для вычислений

Всё это влияет на размер приложения и производительность, что в конечном итоге влияет на взаимодействие с пользователем. При наличии большого количества таблиц, связи могут занимать много памяти. Известно, что уменьшение объема занимаемой памяти на 50% возможно за счёт модификации структура данных, таким образом, дополнительно увеличивая отклик пользовательского интерфейса. Дополнительную информацию см. в разделе Оптимизация.

# Островные таблицы (Data Islands)

**Островная таблица** - это таблица, не связанная с основной моделью данных посредством ключевых полей.

Использование островных таблиц допускается для построения расширенной функциональности, однако, стоит избегать операции объединения (join) данных островных таблиц с данными основной модели данных при реализации вычислений.

Тот же принцип сохраняется при наличии в приложении нескольких множеств таблиц, связанных внутри каждого множества, но не имеющих связей между множествами.

# Большие модели данных

Qlik Sense позволяет обрабатывать очень большие наборы данных. Однако для оптимизации взаимодействия с пользователем и ресурсов (память, процессор, место хранения) есть варианты.  
Рассмотрим следующий пример: есть большой набор данных о заказах в 1 миллиард строк. Необходимо предоставить сводные показатели высокого уровня для руководителей, анализ тенденций для бизнес-аналитиков, а также подробные таблицы и значения для группы обработки заказов. Рассмотрим некоторые варианты реализации.

## Все детали в таблице фактов

В этом случае Qlik Sense выполняет всю работу по отображению подробностей и суммированию показателей от самого низкого уровня детализации до самого высокого необходимого сводного отчета.

* Достоинства - простота. Это самое простое решение с точки зрения кода, достаточно просто связать Заказы на детальном уровне (возможно, на уровне SKU) к модели данных и предусмотреть все показатели высокого уровня, диаграммы тенденций, подробные таблицы и выборки в интерфейсе приложении
* Недостатки - Qlik Sense потребуется агрегировать до 1 миллиарда строк сведений при каждом сделанном выборе. Хоть Qlik Sense и может делать это с приемлемой производительностью, он все же такой подход приведет к более медленному взаимодействию с пользователем, чем это возможно

## Цепочки приложение (chaining)

Подход заключается в создании двух (или более) приложений Qlik, в одном - **детальном приложении**, имеется подробная таблица фактов в качестве основной таблицы фактов, другие - **сводные приложения**, имеют предварительно объединенные (агрегированные) версии таблицы фактов в качестве основных таблиц фактов. Важно, чтобы измерения между приложениями цепочки были одинаковыми. Таким образом, пользователи могут начать работу с аналитикой начиная со сводного приложения, отображающего метрики высокого уровня и диаграммы.

Когда и если пользователю необходимо углубиться в детали, можете использовать функцию передачи выборок из сводного приложения в другое приложение - также сводное, но с более низким уровнем детализации или напрямую в детальное приложение, через URL . Пользователь увидит появление новых диаграмм и вкладок, и (при должном проектировании) ему даже не нужно знать, что он перешел из одного приложение в другое. В итоге, получается, что таблица фактов с 1 миллиардом строк будет использована тогда и только тогда, когда она понадобится пользователю. Остальная часть работы будет выполняться в сводном приложении, которое, например, может быть меньше 100 миллионов строк.

* Преимущества - оптимизирует аппаратное обеспечение и скорость отклика для навигации и построения чартов Qlik Sense. Поскольку выбор пользователей и навигация зависят от их потребностей, ресурс процессора и памяти не тратятся на обработку 1 миллиарда строк данных, когда пользователю не нужны данные, обрабатываемые на самом низком уровне детализации
* Недостатки - таблицы (QVD) необходимо предварительно агрегировать и поддерживать для этого подхода. Хотя это разовая работа по разработке, она немного сложнее, чем вариант 1, где требуется только одна версия таблицы

## ODAG

**ODAG** - это **on-demand application generation**, "генерация приложений по запросу", в процессе чего создаётся (генерируется) "приложение по запросу".

Приложения по запросу предоставляют пользователям агрегированные представления больших данных и позволяют локализовать и загружать соответствующие подмножества данных для подробного анализа.

Приложения по запросу расширяют потенциальные варианты ассоциативного анализа крупных источников данных. Они позволяют пользователям сначала выбирать данные, в которых они заинтересованы, а затем в интерактивном режиме создавать приложение по запросу, с помощью которого они могут анализировать данные с использованием всех возможностей Qlik Sense в памяти.

* Преимущества - в ещё большей степени оптимизирует аппаратное обеспечение и скорость отклика, в частности потому, что нет необходимости загружать в память детальную таблицу в полном объёме
* Недостатки - единственным недостатком можно считать лишь относительно повышенную сложность проектирования, создания и поддержки таких приложений

Подробнее подход описан в [официальной документации](https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DataSource/Manage-big-data.htm).

## Агрегаты и детали в одном приложении

Этот вариант (и ни в коем случае не последний) заключается в использовании предварительно агрегированной сводной таблицы в дополнение к подробной таблице в одной модели данных. Необходимо сформировать и загрузить предварительно агрегированную таблицу как островную таблицу (не связанную с другими таблицами в модели данных). Затем, когда будут сделаны соответствующие выборки в подробной таблице фактов, можно перенести эти выборки в предварительно агрегированную таблицу с помощью анализа множеств и/или альтернативных состояний.

* Преимущества - этот вариант не требует дополнительных приложений и цепочки документов, т.к. использует как детальную, так и сводную версии большой таблицы
* Недостатки - этот вариант потребует некоторых настроек в приложении для срабатывания действий по переносу выборки из одной таблицы в другую. Поскольку приложение со временем меняется, необходимо отслеживать, где и когда применять этот подход

# Оптимизация

Когда фаза разработки приложения завершена и начинается фаза развертывания, очень важно следовать методам оптимизации размера приложения, чтобы работа конечного пользователя была максимально эффективно. В этом разделе рассматриваются методы оптимизации данных и обработки выражений.

* После загрузки удалить все не используемые поля (**DROP FIELDS**) и таблицы (**DROP TABLES**). Ненужное поле - это поле, которое в настоящее время не используется в диаграммах, списках и т.д.
* Для огромных данных, рассмотреть возможность разбить их на несколько временн*ы*х периодов. Например: реализовать отдельно первое приложение "Текущий год к прошлому году", а второе, с данными за 5 лет (или более), так как большинство пользователей нуждаются в сравнительных показателях текущего года по сравнению с прошлым годом

*Примечание: это также приводит к гораздо более быстрому взаимодействию с конечными пользователями. Предположим, что 80% ваших пользователей просматривают данные только за последние 13 месяцев, а приложение содержит данные за 60 месяцев. Если создать две версии приложения (одну который содержит только последние 13 месяцев, а другой – все 60 месяцев), то это позволит пользователям сначала анализировать 13-месячную версию, а к другой версии переходить только при необходимости. Это позволит для этих 80% пользовательских сеансов снизить уровень потребления памяти и процессора, сократить время обработки. Это улучшает работу ваших конечных пользователей и снижает нагрузку на оборудование*

* Не стоит нормализовать данные слишком сильно. Запланируйте от 6 до 10 таблиц в типичном приложении Qlik Sense. Это всего лишь рекомендация, но с моделями данных Qlik Sense необходимо соблюдать баланс
* Избавиться от небольших таблиц-листьев (конечных узлов иерархии модели данных), используя Mapping Load для переноса кодовых значений в другие измерения или таблицы фактов
* Сохранять все, что возможно, в виде числа вместо строки
* Использовать целые числа для объединения таблиц
* Денормализовать таблицы с небольшим количеством полей
* Стремиться максимум к одному уровню измерений от записей фактов в модели "снежинка"
* Используйте автонумерацию **AutoNumber()** для ключевых полей, значения которых не используются в интерфейсе, это сокращает использование памяти; функцию лучше применить в конце скрипта со списком сокращаемых полей
* Использовать шаблон инкрементной загрузки для постепенной загрузки и разбивки QVD-файлов с историей на отдельные файлы на основе добавления периода времени
* Стремиться использовать относительные пути при обращении к файлам
* Включить файлы для подключений
* Разделить timestanp на поля даты и времени, когда требуется дата и время
* Удалить время из даты с помощью **Floor**() или **Date**(**Date**#(..)), когда время не требуется
* (Нет никаких преимуществ при преобразовании буквенно-цифровых полей, когда строка и результирующее числовое поле имеют одинаковую длину)
* Использовать числовые поля в логических функциях (сравнение строк выполняется медленнее)
* Вместо **(a - b) / b** лучше **( a / b ) - 1**
* **Date(Max(sDate, 'DD.MM.YYYY'))** быстрее, чем **Max(Date(sDate, 'DD.MM.YYYY'))**
* Обеспечить необходимый и достаточный для анализа уровень детализации исходных данных - **sum(..) group by**
* Использовать числовые флаги (например, с 1 или 0), которые предварительно вычисляются в скрипте
* Вместо **Sum(If(Flag, Amount))** лучше **Sum(Flag \* Amount)**
* Уменьшите количество одновременно рассчитываемых (отображаемых) объектов
* Стремить к предварительному расчету показателей в рамках скрипта
* Ограничить количество выражений в объектах, по возможности, распределить их по нескольким объектам
* Деактивировать Hyperthreading в BIOS сервера; Hyperthreading (только процессоры Intel) может замедлить обработку скриптов
* Для очень больших приложений можно выполнить дополнительную оптимизацию путем предварительного кэширования выборок в ОЗУ, например, посредством <https://github.com/eapowertools/CacheInitializer>
* Используйте переменные и/или анализ множеств вместо сложных расчетов данных в выражениях