# Руководство разработчика Интеграл

Документ ознакомит читателя с возможностями конструктора Интеграл, а также научит разрабатывать полноценные web–приложения под произвольные требования заказчика.

Задача Интеграла – избавить пользователя от необходимости прикладного программирования. Это среда разработки приложений, а сама разработка сводится к постановке задачи – пользователь описывает модель данных, правила их обработки и формат выдачи результатов. Конструктору Интеграл меньше года, в нем реализованы только основные возможности, но он постоянно развивается, чтобы стать более понятным и дружественным к пользователю.

Ссылка на следующий раздел [для нетерпеливых](#_Регистрация,_установка_и), кто не любит много читать.

Автором приветствуются любые замечания и предложения по функционалу системы и по данному руководству. Пишите: [admin@bkint.ru](mailto:admin@bkint.ru).

[Руководство разработчика Интеграл 1](#_Toc480529035)

[Регистрация, установка и состав системы 2](#_Toc480529036)

[Архитектура приложения Интеграл 2](#_Toc480529037)

[Модель данных 3](#_Toc480529038)

[Редактор типов Интеграла 4](#_Toc480529039)

[Значения по умолчанию и проверка данных 8](#_Toc480529040)

[Валидация (проверка значений) данных 8](#_Toc480529041)

[Вычисляемые реквизиты 9](#_Toc480529042)

[Справочник 9](#_Toc480529043)

[Поиск и редактирование 10](#_Toc480529044)

[Экспорт и импорт данных 12](#_Toc480529045)

[Отчеты 14](#_Toc480529046)

[Агрегирование в отчетах 17](#_Toc480529047)

[Интерактивные отчеты 17](#_Toc480529048)

[Фильтрование данных в отчете и экспорт в MS Excel 18](#_Toc480529049)

[Использование отчетов в Редакторе типов 18](#_Toc480529050)

[Колонки отчета: подробнее 21](#_Toc480529051)

[Использование вложенных запросов в отчете 22](#_Toc480529052)

[Вызов отчета с произвольными параметрами 23](#_Toc480529053)

[Запросы на изменение данных 24](#_Toc480529054)

[Выгрузка отчета в формат JSON 27](#_Toc480529055)

[Реквизит типа Кнопка (BUTTON) 28](#_Toc480529056)

[Разграничение прав пользователей 28](#_Toc480529057)

[Пользовательские файлы 30](#_Toc480529058)

[Генерация web–страниц Интеграла 31](#_Toc480529059)

[Пользовательские html–формы и блоки данных 31](#_Toc480529060)

[Служебные слова, функции и константы 35](#_Toc480529061)

## Регистрация, установка и состав системы

Установка и выбор шаблона для Интеграла здесь: <https://bkint.ru/register.html>

Следующая ссылка [для нетерпеливых](#_Бизнес–модель).

После установки вы получаете доступ к собственной базе, созданной по одному из бизнес–шаблонов: Фитнес–клуб, Розничная торговля и т. п. Шаблон содержит также инструкцию пользователя для этой конкретной реализации базы данных.

Пользователь, созданный при регистрации вместе с базой, имеет имя, совпадающее с именем базы, и права администратора в системе.

## Архитектура приложения Интеграл

Система состоит из пяти универсальных форм:

1. Форма редактирования типов (меню Типы)
2. Словарь (справочник) типов объектов (меню Словарь)
3. Список объектов выбранного типа (любого значения в меню Словарь)
4. Форма редактирования объектов
5. Форма отчета

Модель данных доступна вам через меню «Типы», список всех независимых типов (тех, что не являются реквизитами других типов) находится в меню «Словарь».

Часть функционала шаблона может быть сделана при помощи пользовательских html–форм и отчетов. Список html и css файлов находится в меню «Файлы» в директории templates. В директории download меню «Файлы» хранятся файлы, загруженные пользователями базы данных.

Внимание: Система не поддерживает запуск пользовательских исполняемых файлов (.php, .asp и т. д.).

Вы можете изменить и дополнить модель данных под свои нужды (или под нужды вашего заказчика), создать или отредактировать html–формы и обновить инструкцию пользователя в соответствии с внесенными изменениями.

**Основной принцип Интеграла: его программист не обязан знать язык программирования или теорию баз данных, а максимальная требуемая квалификация не превышает уровень пользователя MS Excel. Интеграл позволяет решать любую задачу, оперируя только терминами бизнеса и не неся никаких накладных расходов при постановке задачи и ее выполнении.**

Разумеется, самыми примитивными функциями уровня Excel (СУММА, СРЕДНЕЕ и т.п.) познакомиться, вероятнее всего, понадобится, что не противоречит озвученному выше принципу. Также будет очень полезно знать список [служебных слов](#_Служебные_слова_и) Интеграла и правила их использования.

Значительная часть функционала Интеграла построена средствами самого Интеграла, это Пользователи, Роли пользователей, Объекты доступа и Отчеты. Вы можете использовать и дополнять соответствующие сущности, исходя из ваших нужд, но при этом следует соблюдать осторожность. Категорически не рекомендуется удалять перечисленные типы или их реквизиты.

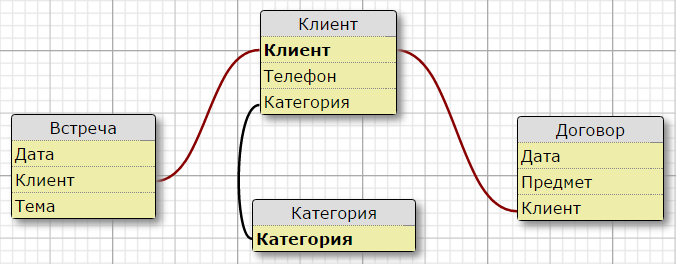
## Модель данных

В вашем распоряжении модель данных, состоящая из связанных между собой таблиц. Вы определяете термины, такие как Клиент, Счет, Договор, Телефон, Встреча, Категория, а затем указываете связи между соответствующими объектами. С помощью этих связей система самостоятельно агрегирует данные для навигации в системе и при построении отчетов. Система позволяет создать любую иерархию данных, подобно реляционной базе данных.

Например:

* Клиент имеет реквизит Телефон (подчиненный реквизит Телефон);
* Клиент имеет списочный реквизит Категория (подчиненный реквизит, ссылающийся на значение из справочника Категорий);
* Для Клиента хранится информация обо всех проведенных с ним Встречах (подчиненный массив Встречи);
* Договор имеет ссылку на Клиента, с которым он заключен (Один клиент — много Договоров).

Структура базы данных для описанного выше примера:



Следующая ссылка [для нетерпеливых](#_Редактор_типов_Интеграла).

Все эти данные представлены в Интеграле в виде таблиц. Таблица Категория:

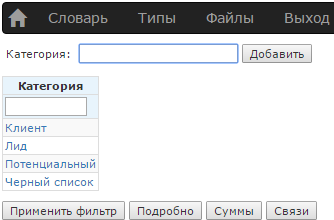
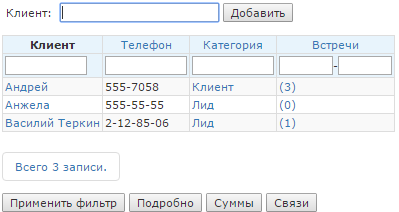
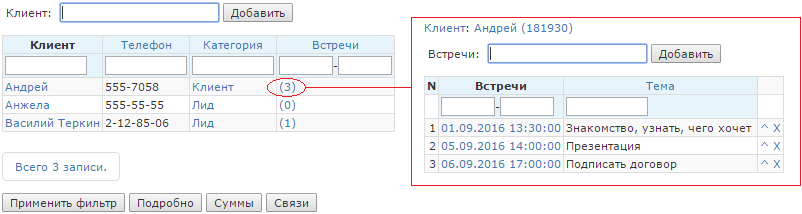


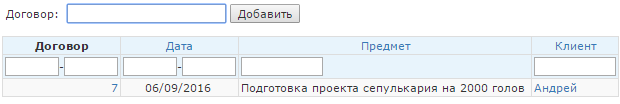
Таблица Клиент с реквизитами Телефон, Категория (из списка категорий), Встречи (массив):

****

Нажав на значение в колонке «Встречи» можно посмотреть связанный массив данных по встречам этого клиента:



Список договоров в системе содержит ссылки на клиентов, с которыми заключен договор:

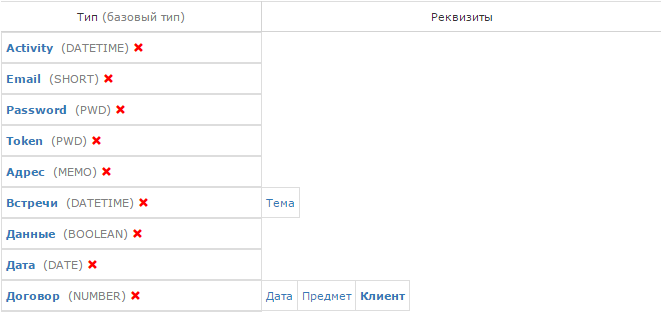


## Редактор типов Интеграла

Редактор типов находится в меню Типы. Вы обнаружите, что в Редакторе типов уже зарегистрировано какое-то множество объектов, потому что Интеграл написан, по большей части, средствами самого Интеграла, что позволяет вам дописывать и переписывать его под себя.

Типы имеют определенный базовый тип и могут иметь реквизиты, которые будут перечислены в колонке «Реквизиты»:

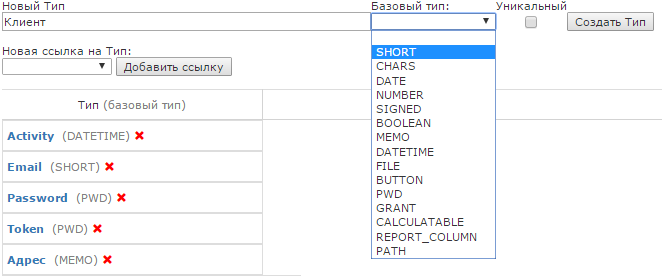


Все термины (Типы), используемые системой, создаются здесь (см. *Новый тип –> Создать Тип* на рисунке). Вы задаете название Типа данных и указываете для него *базовый тип* данных:

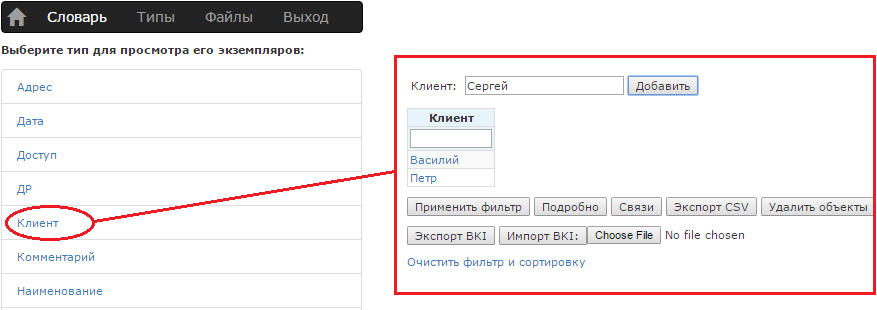
|  |  |
| --- | --- |
| **Базовый тип** | **Описание** |
| SHORT | Короткая строка (до 127 символов) |
| CHARS | Строка неограниченной длины |
| MEMO | Текст (отличается от CHARS только полем ввода – оно многострочное) |
| DATE | Дата |
| DATETIME | Дата и время |
| FILE | Файл |
| NUMBER | Целое число |
| SIGNED | Число с десятичной частью |
| BUTTON | Действие (кнопка) над экземпляром типа |
| BOOLEAN | Логическое значение (Да / Нет) |

Базовый тип сообщает системе, как выводить и обрабатывать соответствующие значения.

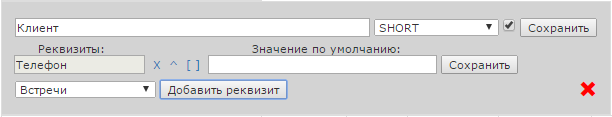
Для добавления типов в Интеграл необходимо задать имя нового типа, выбрать ему базовый тип и нажать «Создать тип». Для примера создадим тип «Клиент»:



Как только вы добавили термин, в меню Словарь появляется соответствующий пункт, и вы сразу же можете заносить в систему объекты этого типа. Попробуйте:



Аналогичным образом создаются все остальные термины (типы): Счет, Договор, Телефон, Встреча, Категория. После этого можно подчинять одни типы другим, создавая сущности произвольного состава: выбираем тип из списка и нажимаем «Добавить реквизит».

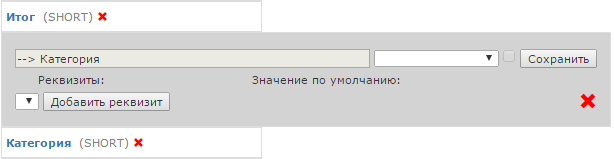


Все изменения в справочнике типов немедленно вступают в силу и отображаются в Словаре.

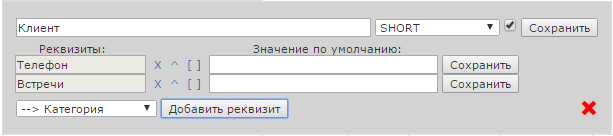
Если какой-то тип планируется использовать в качестве справочного значения, то необходимо создать ссылку на этот тип, которая затем будет использоваться в качестве реквизита. Например, добавим ссылку на тип Категория:



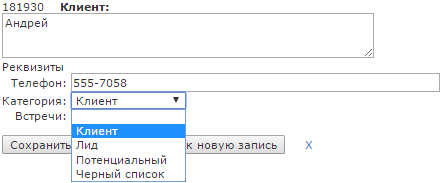
В списке типов появится новый тип с префиксом ссылки: «--> Категория».



А теперь добавим соответствующий реквизит клиенту:

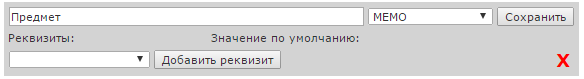
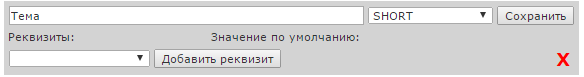
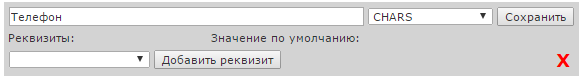
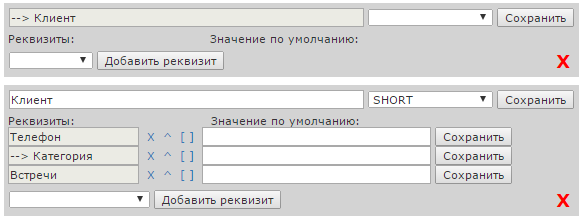
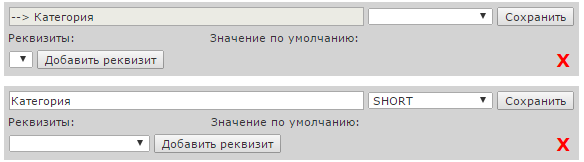
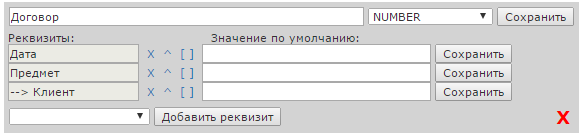
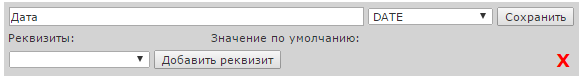
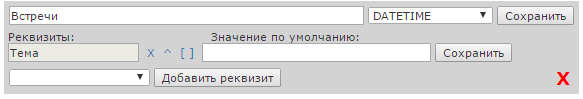


Во время редактирования такого реквизита можно выбрать значение из списка:

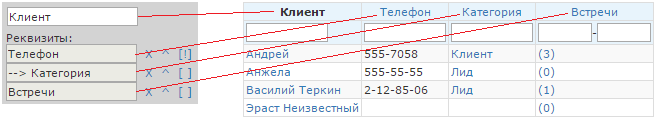


Реквизиты можно добавлять, удалять (знак «Х» напротив реквизита) и изменять их порядок (нажатие на знак «^» перемещает реквизит на одну позицию выше).

Развернутое описание рассмотренной [выше](#_Бизнес–модель) структуры данных в Редакторе типов выглядит так:



Последовательность реквизитов в Редакторе определяет их последовательность и в таблицах системы:



В первой колонке таблицы содержится значение объекта, в остальных – его реквизиты. Значение также всегда имеет уникальный системный идентификатор (цифровой, выделен на рисунке красным):



Обычному пользователю этот идентификатор, вероятнее всего, никогда не понадобится. Однако, как мы увидим позже, знание о нем может пригодиться аналитику при настройке системы.

Обратите внимание, что если реквизит сам имеет реквизиты, то он отображается в виде массива данных, то есть позволяет задать не одно, а несколько значений (см. реквизит *Встречи* типа *Клиент*). Такой подход в некоторых случаях может значительно упростить навигацию в системе.

 Нетерпеливым [ссылка](#_Справочник).

**Описанные возможности редактора уже позволяют вам создать сколько угодно сложную модель данных. При этом нет необходимости в каких–либо дополнительных знаниях о базах данных: внешние ключи, индексы и т.д., Интеграл позаботится об этом за вас. Просто задайте нужные вам таблицы, и пользователи сразу смогут начинать работать с ними.**

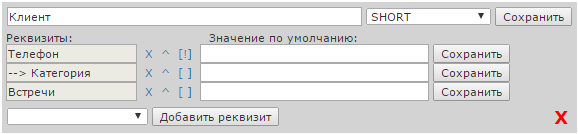
**Вы можете в любой момент добавлять новые объекты и связи, и они немедленно будут доступны пользователям.**

## Значения по умолчанию и проверка данных

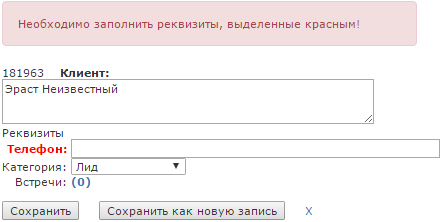
*Данный раздел будет постоянно развиваться: в ближайшее время добавится проверка значений реквизитов по маске (regexp), средства задания формата и многое другое.*

### Валидация (проверка значений) данных

Одно из простейших средств для проверки данных при вводе – это реквизит, обязательный для заполнения. Вы можете указать его в Редакторе типов, нажав на квадратные скобки напротив выбранного реквизита (в квадратных скобках появится знак «!»):



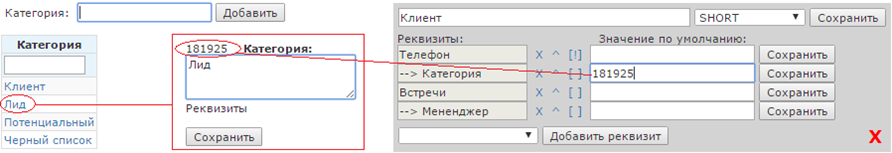
При попытке сохранить неверно заполненную форму система выведет предупреждение и оставит пользователя в режиме редактирования формы:



### Вычисляемые реквизиты

Поле «Значение по умолчанию» можно также использовать для автоматического заполнения значений реквизитов при создании объекта: то, что вы впишете в это поле, попадет в соответствующее поле формы редактирования при создании новой записи. В этом поле также можно использовать [служебные слова](#_Служебные_слова_и).

Для ссылочных реквизитов («--> Клиент», «--> Категория») следует указывать не имя Клиента или название Категории, а их внутренний идентификатор (ID): система хранит ссылки на ID (имя может быть изменено, а идентификатор – нет). Например, если мы хотим по умолчанию присваивать клиенту категорию «Лид», то нам нужно прописать для поля «--> Категория» значение его ID:



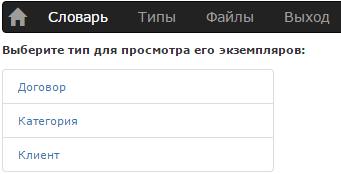
Не забудьте нажать кнопку «Сохранить» или Enter после редактирования каждого поля!

Теперь при создании клиента система сама проставит ему Категорию «Лид», причем пользователь сможет изменить это значение на любое другое.

Если объект с идентификатором, вычисленным по умолчанию, не соответствует типу реквизита, то его значение не будет использовано и не будет сохранено в базу формой редактирования объекта.

## Справочник

Список всех независимых типов (тех, что не являются реквизитами других типов) находится в меню «Словарь»:

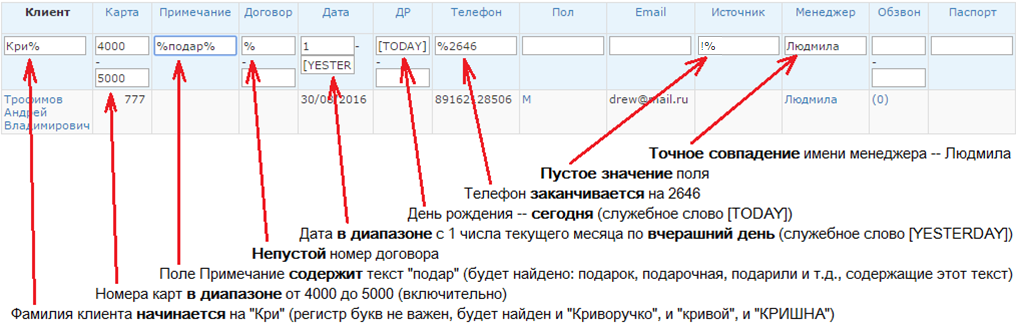


Администратор системы может настроить доступ пользователей только к определенным словарным типам, используя механизм [ролей](#_Разграничение_прав_пользователей).

## Поиск и редактирование

По умолчанию в таблицах системы отображаются только первые 20 записей, а поиск объектов осуществляется с помощью фильтров в заголовке таблицы. Поля фильтра открываются по клику на шапке таблицы объектов или по нажатию иконки «воронка» над шапкой таблицы. Фильтр может быть диапазонным, если это потенциально применимо (для дат и чисел), или по маске.

Типичные способы задания фильтров показаны на рисунке:



Обратите внимание на конструкцию !% («не непустой», в противоположность %). Эта конструкция одинаково работает и в списках, и в отчетах. Вообще, модификатор «!», используемый в начале строки условия, обозначает отрицание последующего условия, например, !%подар% будет означать «всё, что не содержит ‘подар’, включая пустые (отсутствующие) значения».

Вообще, модификатор «!», используемый в начале строки условия, обозначает отрицание последующего условия, например, !%подар% будет означать «всё, что не содержит ‘подар’, включая пустые (отсутствующие) значения».

Обратите внимание, что при задании даты в текущем месяце текущего года можно указать только число. Например, если сегодня 31.08.2016, то:

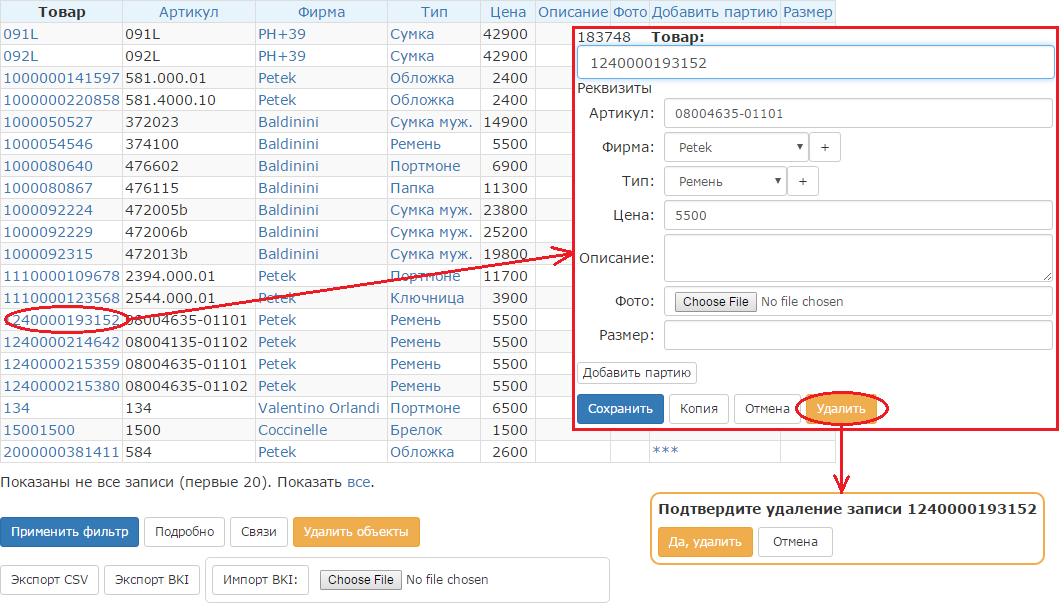
12 – 12-е число текущего месяца: 12.08.2016

12.6 – 12 июня текущего года: 12.06.2016

При указании года двумя цифрами, например 12.12.16, система сохранит год в текущем веке: 12.12.2016. Если необходимо указать год в другом веке, то его нужно указывать целиком: 12.12.1916.

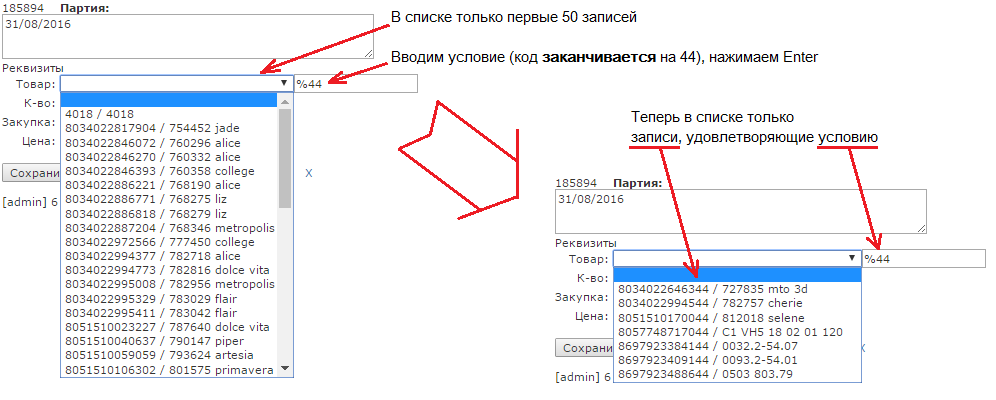
В качестве разделителя дат можно использовать символы «.», «,» или «/».

Чтобы отредактировать запись, нужно нажать на значение в ее первой колонке:

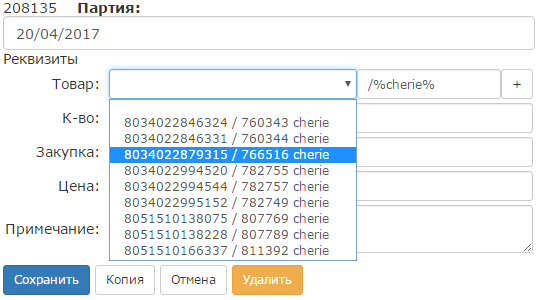


Удалить запись можно нажатием кнопки «Удалить», при этом Интеграл запросит подтверждение на удаление. Если на запись есть ссылки в системе (например, это [списочное](#Ссылка_на_тип) значение, использующееся в объектах Интеграла), то система выдаст предупреждение и не позволит удалить эту запись.

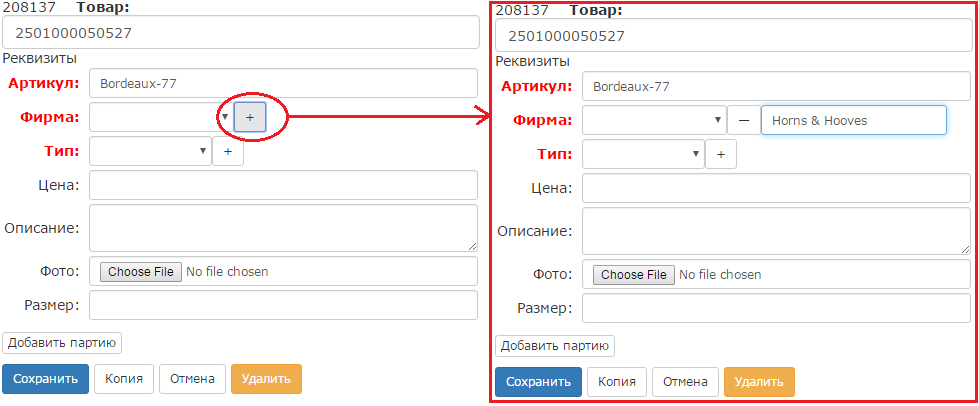
Если в выпадающем списке какой–то формы более 50 записей, то в списке присутствуют только первые 50 из них, и тогда для поиска записей нужно использовать поле справа от списка:



Для поиска по различным полям выпадающего списка следует разделять маски поиска знаком «слэш», например, для поиска записей со словом «cherie» во второй колонке нужно использовать условие поиска: /%cherie% (или %/%cherie%):

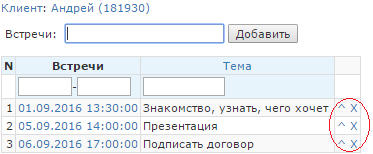


Если у пользователя есть доступ для создания объектов используемого списочного значения, например, Товар на рисунке выше, то слева от списка будет изображена кнопка немедленного добавления списочного значения (со знаком «+»). По этой кнопке можно задать новое значение, которое будет добавлено в список при сохранении формы:



Система позволяет сделать копию любого объекта вместе со всеми его реквизитами, включая массивы. Для этого нужно открыть объект, при необходимости внести изменения и нажать кнопку «Копия».

При добавлении записей в подчиненный массив данных они сохраняются в порядке добавления. Записи можно удалять (значок «Х») или перемещать (значок «^» перемещает запись на одну позицию вверх):



## Экспорт и импорт данных

Пользователь может экспортировать структуры данных и сами данные, а затем импортировать их в другую базу Интеграла или использовать по другому назначению. Также возможна подготовка данных из сторонних систем для загрузки их в Интеграл.

Формат выгрузки содержит необходимый минимум информации, начиная с описания типа, для которого делается выгрузка.

Описание типов (после описания каждого типа идет описание подчиненных и связанных типов):

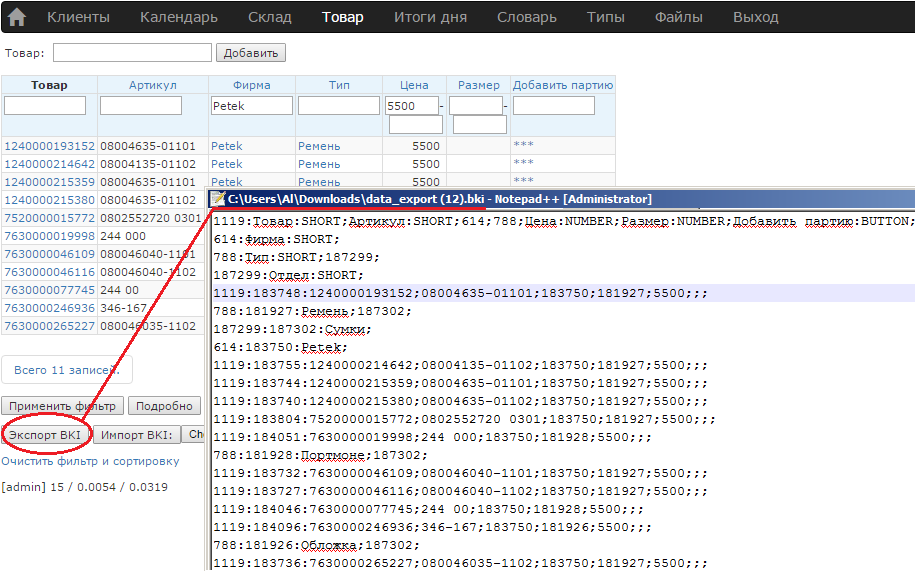
**ID\_типа:имя\_типа:базовый\_тип\_типа;[имя\_реквизита:базовый\_тип\_реквизита; | ID\_типа\_ссылочного\_реквизита;, ...]**

Описание данных:

**ID\_типа\_записи:[ID\_записи];значение\_записи;[реквизит;, ...]**

Для экспорта необходимо зайти в нужный вам справочник, отфильтровать данные, если требуется, и нажать кнопку «Экспорт BKI». Будет выгружен файл, содержащий:

1. Описание структуры данных, включая все подчиненные данные и связанные справочники
2. Сами данные (на рисунке ниже данные начинаются с выделенной серым строки)



Таким образом вы можете:

* экспортировать ваши наработки, например, если вы хотите продать разработанные вами шаблоны новому клиенту;
* экспортировать и импортировать справочники, включая иерархические структуры любой вложенности;
* импортировать подчиненные массивы данных для выбранного объекта (например, историю встреч клиента).

Интеграл корректно разрешит ситуацию, если в целевой системе уже есть нужные вам типы данных: они будут найдены по комбинации «Тип:базовый тип» и использованы в качестве терминов или их реквизитов.

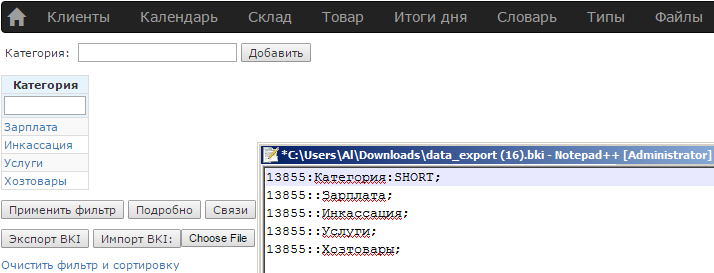
Интеграл разрешит проблемы с конфликтующими ID в целевой системе, но прекратит импорт, если реквизит с данным номером уже занят, и существующий реквизит имеет имя и тип, отличные от загружаемого. Например, Интеграл сообщит об ошибке, если вы импортируете Клиентов с двумя реквизитами Имя (SHORT) и Телефон (SHORT), а в целевой базе Клиент имеет реквизит Имя с типом CHARS или эти реквизиты имеют другой порядок или названия (например, «Тел.» вместо «Телефон»). Если же все имеющиеся реквизиты совпадают, но вы импортируете какие-то дополнительные, то они будут добавлены к описанию Типа в целевой системе. Например, вы можете выгрузить Клиентов с указанием Имени и Телефона, добавить в выбранный файл третий реквизит Категория, указать значение этого реквизита для всех клиентов и загрузить всё это обратно в Интеграл (предварительно удалив выгруженных клиентов из справочника!).

Если выгружаемые значения могут быть использованы как справочные (в Справочнике типов есть реквизит ссылки на этот Тип, например, «--> Товар»), то данные будут выгружены вместе с идентификаторами. В противном случае идентификаторы выгружены не будут, поскольку не имеют критичного значения.

Если в каких-то полях данных встретятся переводы строки или точки с запятой (служащие разделителями в файле), то Интеграл с этим легко справится и корректно загрузит данные, экранировав точки с запятой и контролируя количество полей данных.

Если в загружаемом файле указаны ID записей, то Интеграл не будет заменять уже имеющиеся записи с такими ID, значением и типом, не проверяя соответствие остальных реквизитов. Если ID занят, а тип и значение не совпадают, то Интеграл прекратит импорт, сообщив о конфликте (ID занят объектом с другим значением или типом).

Формат выгрузки или загрузки для простейшего справочника, который можно выгрузить из Интеграла или подготовить вручную приведен на рисунке:



## Отчеты

Построитель отчетов Интеграла позволяет конструировать отчет, просто выбирая нужные пользователю колонки и не заботясь о том, как связать таблицы, участвующие в построении отчета.

**Построитель отчетов позволяет реализовать любую конструкцию языка SQL, включая вложенные запросы. При этом сам язык SQL знать не обязательно.**

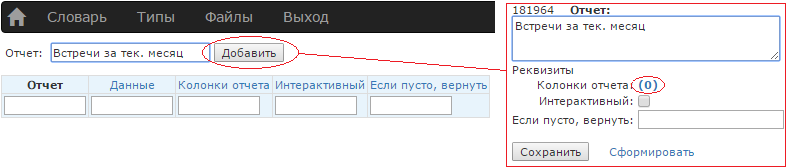
Допустим, нам требуется выбрать всех клиентов, с которыми были подписаны договоры и прошли встречи в этом месяце, в формате:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Клиент | Категория | Телефон | Дата | Тема встречи |

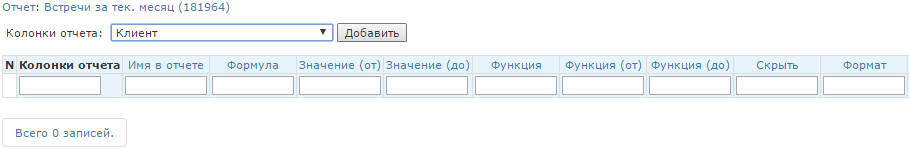
Также нужно подсчитать общее количество таких встреч.

Такой отчет потребует выборки данных из 4 таблиц системы, но это нас нисколько не заботит.

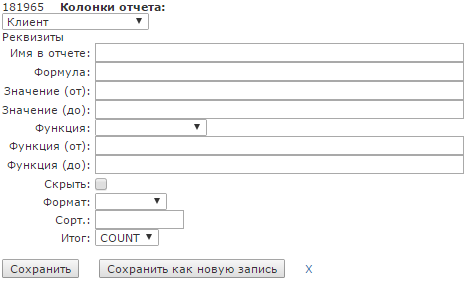
Заходим в меню *Словарь* –> *Отчет* и добавляем нужный нам отчет:



Заходим в массив *Колонки отчета* (см. красный овал справа на предыдущем рисунке) и добавляем колонку «Клиент»:

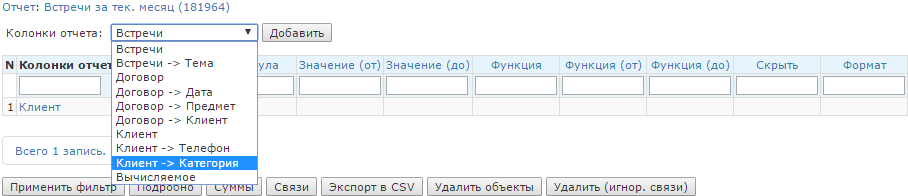


На открывшейся форме выбираем «COUNT» в выпадающем списке «Итог» и нажимаем «Сохранить»:

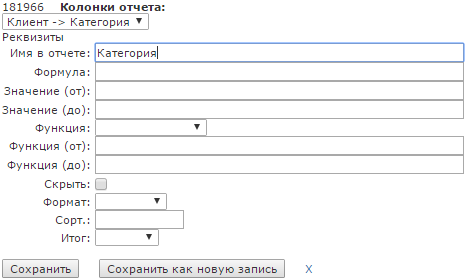


Графа «Итог» служит для указания функции, которую нужно вывести в итоговой строке отчета по этому полю: AVG (среднее), COUNT (количество), MAX (максимальное), MIN (минимальное), SUM (сумма).

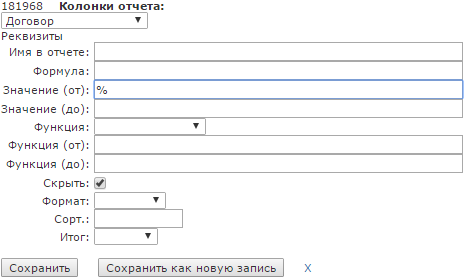
После того, как мы добавили первую колонку отчета, выпадающий список «Колонки отчета» сократился до тех значений, которые имеют связь с таблицей клиентов:



Добавляем колонку «Клиент –> Категория», правим поле *Имя в отчете* на «Категория» и сохраняем:

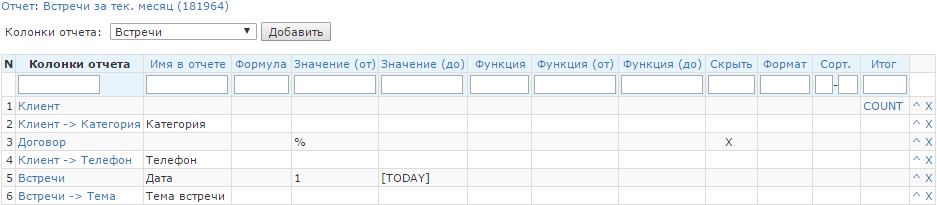


Теперь в нашем отчете уже 2 колонки, и мы добавляем третью: «Договор». Сама по себе эта колонка в отчете не нужна, но у нас есть условие, что клиент должен попасть в отчет, только если у нас с ним есть договор. Здесь мы задаем условие, что поле «Договор» не должно быть пустым. В полях «Значение (от)» и «Значение (до)» используются те же правила, что рассмотрены в [Поиск и редактирование](#_Поиск_и_редактирование):

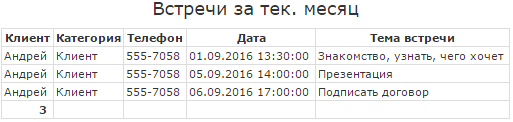


Обратите внимание на поставленную галку «Скрыть»: эта колонка не будет отображена в отчете.

Аналогичным образом добавляем остальные колонки в отчет, не забыв указать условия по дате встречи: с 1 по сегодняшнее число текущего месяца (см. колонку «Встречи»):

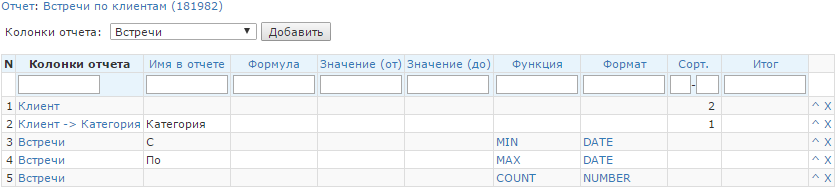


Все эти действия заняли у нас не больше 2 минут, и теперь по кнопке «Сформировать» на форме отчета мы получим такой отчет, построенный точно по нашему заданию:

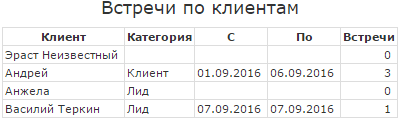


### Агрегирование в отчетах

Также в отчете можно применять агрегирующие функции: сумма, количество, минимальное, максимальное, среднее. В следующем примере создан отчет, который выводит количество встреч с клиентами с указанием периода, в который эти встречи прошли:

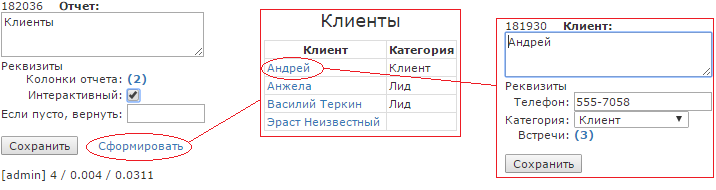


Отчет отсортирован сначала по категории клиента, затем по его имени. Также, для удобства восприятия, формат даты приведен к виду DATE (дата), вместо DATETIME (дата и время):



### Интерактивные отчеты

Если в отчете выставлена опция «Интерактивный», то объекты в этом отчете отображаются гиперссылками, по которым можно попасть на форму просмотра и редактирования этого объекта:

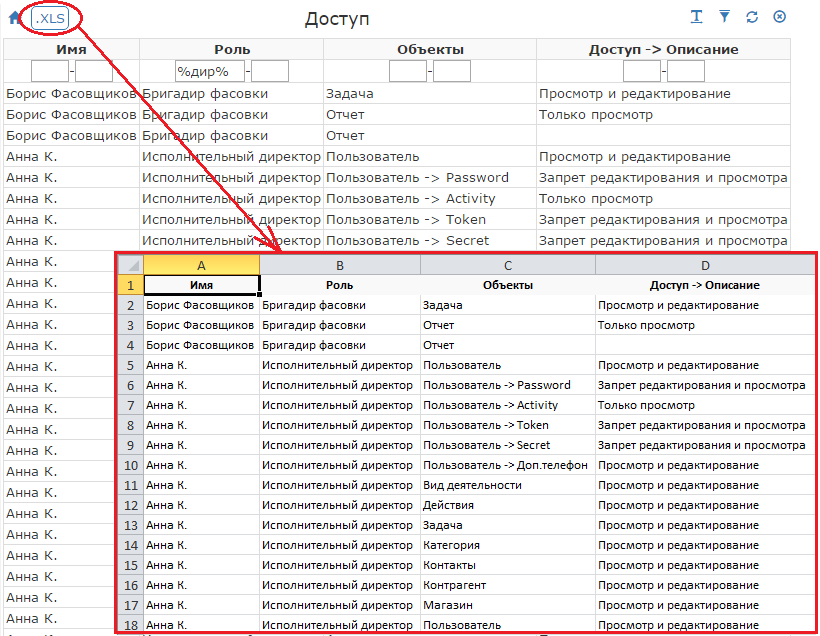


**Вы, вероятно, уже подумали, что Редактор типов и построитель отчетов дают такую комбинацию возможностей, которой нет ни в одном другом средстве разработки. Вы совершенно правы. Это еще не всё, что может Интеграл. И, кроме того, он постоянно развивается.**

### Фильтрование данных в отчете и экспорт в MS Excel

Интеграл позволяет применять фильтры к данным в уже построенном отчете, при этом отчет будет перезапрошен с сервера, чтобы удовлетворять заданным условиям. Фильтр открывается по клику на шапке отчета или иконке «воронка» справа вверху отчета.

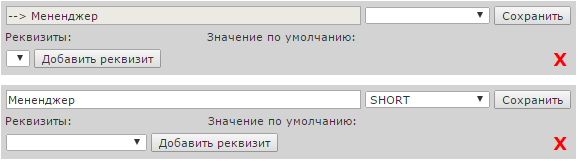
Сформированный отчет можно выгрузить в MS Excel соответствующей кнопкой.



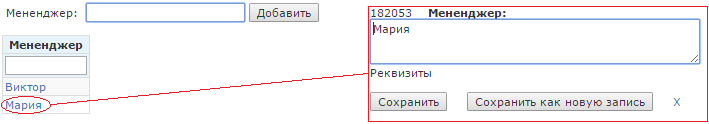
### Использование отчетов в Редакторе типов

С помощью отчетов мы можем вычислять поля по умолчанию на формах редактирования объектов. Например, при заведении в компьютер нового клиента нам нужно проставить имя менеджера, который его завел.

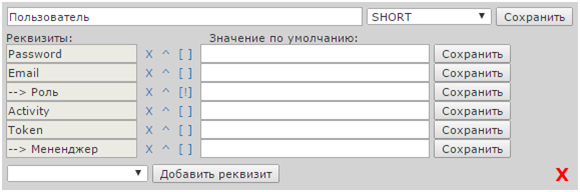
Добавим в Редакторе типов новый тип *Менеджер* и ссылку на него:

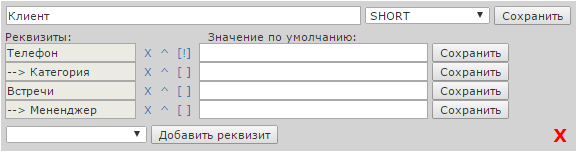


Добавим наших менеджеров в таблицу (Словарь –> Менеджер):

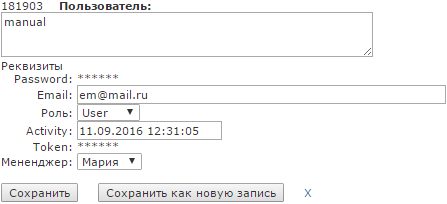


Затем добавим ссылку на менеджера в таблицы *Пользователь* и *Клиент* в Редакторе типов:

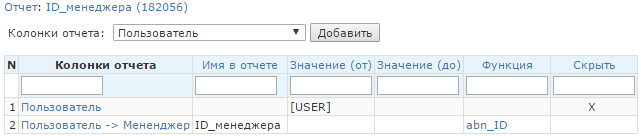




Теперь мы можем указать для каждого пользователя, какой менеджер под ним работает:

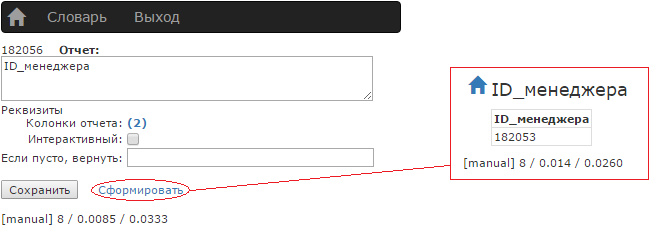


Далее мы сделаем отчет (назовем его «ID\_менеджера»), который позволит нам вычислить менеджера по имени пользователя:

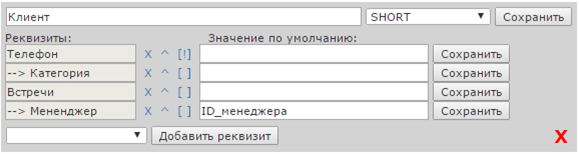


Здесь мы используем фильтр для поля пользователь: служебное слово [USER], а также служебную функцию abn\_ID, чтобы преобразовать имя менеджера в его идентификатор (список служебных слов смотрите в разделе [Служебные слова и функции](#_Служебные_слова_и)).

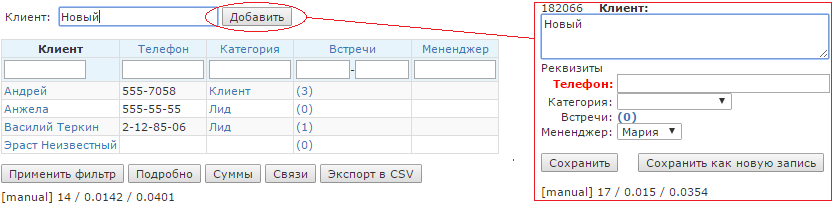
Когда мы запустим этот отчет, то увидим в нем единственную строку с идентификатором менеджера *Мария* (182053), привязанного нами ранее к пользователю *manual*:



В поле «Значение по умолчанию» реквизита «--> Менеджер» (ссылка на менеджера) типа *Клиент* проставим имя нашего отчета: *ID\_менеджера*.



Теперь Интеграл будет знать, что при создании нового Клиента нужно запустить отчет *ID\_менеджера* и подставить полученное в нем значение в поле Менеджер:



**Обратите внимание, с какой скоростью и легкостью мы обогатили функционал системы: создали новый термин, добавили колонки в таблицы, построили отчет с учетом новых связей и добавили правила вычисления реквизитов. Мы не останавливали работу других пользователей, ничего не программировали и ни к кому не обращались за помощью.**

В Интеграле всё это заняло не более 5 минут, включая тестирование доработок: мы описали, чего хотим, и это сразу заработало. В любой другой системе максимум, что вы можете сделать без программиста, это добавить колонки и таблицы.

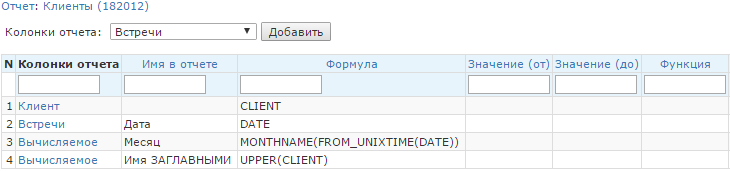
## Колонки отчета: подробнее

Рассмотрим подробнее назначение полей Колонки отчета:

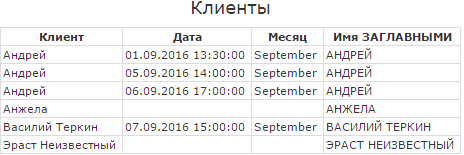
**Имя в отчете** – название колонки в выводимом отчете, по умолчанию – значение Колонки отчета.

**Формула** задает вычисляемое выражение или псевдоним колонки. В выпадающем списке колонок отчета на последней позиции находится значение «Вычисляемое». Это синтетическое поле, которое позволяет вставить произвольные операторы и функции, допустимые в языке SQL. При этом можно оперировать любыми другими колонками отчета, если для них задан псевдоним в их поле *Формула*.

Например, следующий отчет вычисляет название месяца на английском из даты встречи (в базе она хранится в секундах с 01-01-1970 00:00:00) и преобразует имя клиента в заглавные буквы:



Отчет при этом будет выглядеть так:



Вряд ли вы будете часто использовать подобные возможности, но вы должны знать, что Интеграл предоставляет доступ к очень мощным и гибким средствам для различных вычислений.

Если в поле Формула встречается выражение в квадратных скобках, то система попытается найти соответствующий отчет и использовать данные из него (см. [Использование вложенных запросов в отчете](#_Использование_вложенных_запросов)).

**Значение (от), Значение (до)** – диапазон значений (для дат и чисел) или маска (для текста). Здесь работают правила фильтрования из раздела [Поиск и редактирование](#_Поиск_и_редактирование) и можно использовать [служебные слова](#_Служебные_слова). Также здесь можно использовать значение других полей отчета, если заключить их имя из колонки *Формула* в квадратные скобки.

**Функция** – агрегирующая функция, применяемая к значению поля: AVG (среднее), COUNT (количество), MAX (максимальное), MIN (минимальное), SUM (сумма). Группировка значений, посчитанных этой функцией, происходит по всем полям, в которых не используется агрегирование (не указана одна из перечисленных функций): в рассмотренном выше отчете «Встречи по клиента» группировка осуществляется для всех комбинаций Клиент – Категория.

**Функция (от), Функция (до)** – диапазон значений (для дат и чисел) или маска (для текста), применяемые *после* вычисления функций, указанных в поле *Функция* (аналог ключа HAVING в SQL). Здесь также работают правила фильтрования из раздела [Поиск и редактирование](#_Поиск_и_редактирование) и можно использовать [служебные слова](#_Служебные_слова). Также здесь можно использовать значение других полей отчета, если заключить их имя из колонки *Формула* в квадратные скобки.

**Скрыть** – если поставить галку в этом поле, то соответствующая колонка не будет отображена в отчете, хотя ее значение может использоваться для вычислений, фильтрования и сортировки отчета.

**Формат** – конечный формат отображения поля. Служит для приведения значения к нужному виду.

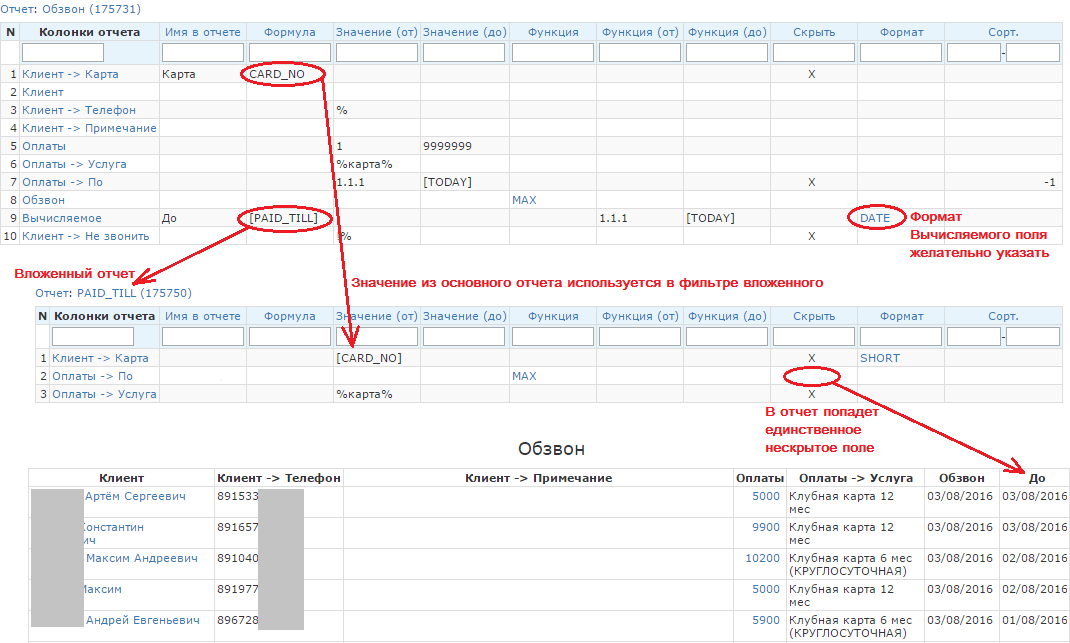
**Сорт.** – указатель последовательности и направления сортировки отчета. Отрицательное число здесь даст обратную сортировку (по убыванию значений).

**Итог** – агрегирующая функция, применяемая к значениям колонки отчета. Если указана функция хотя бы в одной колонке, то в отчете выводится итоговая строка с соответствующими значениями.

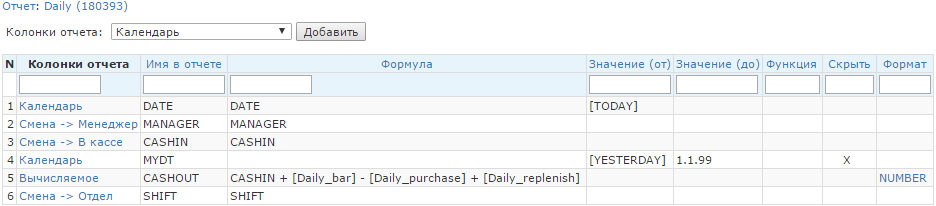
## Использование вложенных запросов в отчете

Приведем более сложный пример для построения отчета. Задача: у клиента есть несколько оплаченных услуг, и нам необходимо выбрать тех клиентов, у кого оплаченное время скоро заканчивается. При этом нужно игнорировать клиентов, кто уже заранее оплатил аналогичную услугу. Получается, нужно дважды обратиться к таблице Оплаты с разными запросами.

В 9-ой строке Колонок отчета используется вызов отчета PAID\_TILL, который в качестве условия отбора использует номер карты CARD\_NO исходного отчета: отчет PAID\_TILL выберет максимальную дату, до которой оплачена услуга по заданной карте.



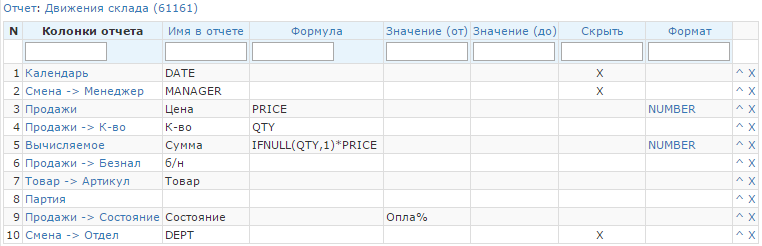
Еще один пример использования вложенных отчетов в поле *Формула*: данные собираются из трех дополнительных отчетов Daily\_bar, Daily\_purchase и Daily\_replenish. Подобные решения применяются, например, при расчете оборотно–сальдовой ведомости.



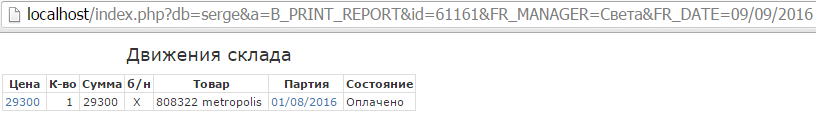
## Вызов отчета с произвольными параметрами

Пользователь может передать в отчет свои параметры для фильтрации данных.

Например, есть такой отчет:



Он выводит следующую информацию:



Обратите внимание на параметры в строке адреса: FR\_MANAGER=Света&FR\_DATE=09/09/2016

Эти параметры позволяют передать в отчет произвольные фильтры, которые будут применены построителем отчета. Правила таковы: для фильтра «Значение (от)» колонки отчета MANAGER следует передать в отчет FR\_MANAGER=значение; для фильтра «Значение (до)» используется конструкция TO\_MANAGER=значение. Фильтр можно применить только к тем колонкам, для которых задано «Имя в отчете».

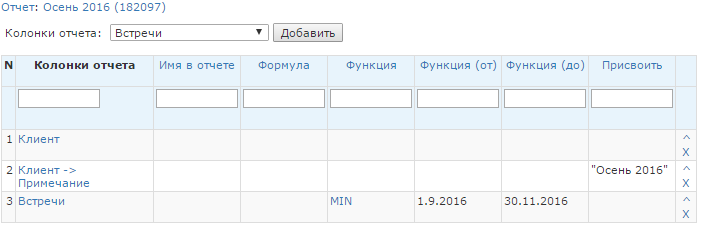
Вам ни в коем случае не следует привыкать задавать эти параметры вручную через строку адреса, но вы можете передать их в качестве параметра из собственной html–формы, как описано в разделе [Пользовательские html–формы](#_Пользовательские_html–формы).

## Запросы на изменение данных

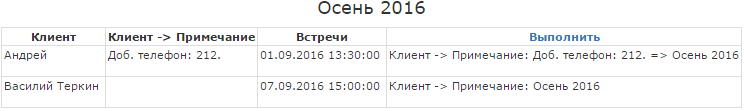
Интеграл позволяет массово изменять данные, в том числе находящиеся в разных таблицах. Запрос на изменение можно добавить к любому отчету – необходимо указать новые значения колонок, отобранных в отчете. Таким запросом можно удалять объекты и их реквизиты, а также создавать новые записи, подчиненные или связанные с выбранными в отчете объектами.

Изменение данных происходит в 2 этапа. Сначала формируется отчет, в котором указаны какие данные будут изменены и как, после чего пользователь дает команду выполнить эти изменения, и они выполняются.

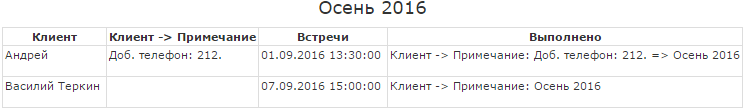
Рассмотрим такой пример: нам необходимо отобрать всех Клиентов, с кем первая встреча была осенью 2016 года, и указать в Примечании Клиента «Осень 2016». Отчет, выполняющий эту работу, будет выглядеть так (обратите внимание на графу «Присвоить»):



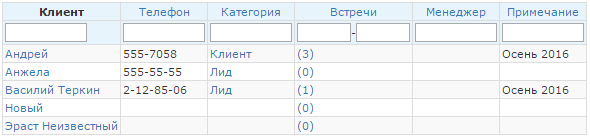
Результат работы отчета будет следующим:



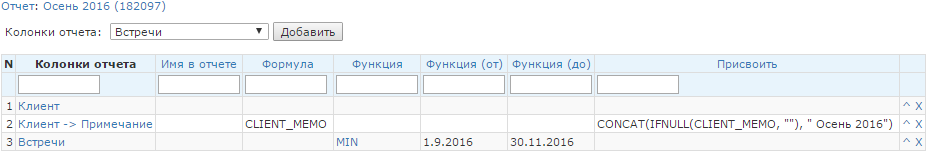
В графе Присвоить указано, какие изменения будут внесены в данные. При нажатии гиперссылки «Выполнить» отчет сформируется еще раз, и все изменения будут применены. Отчет при этом будет отображать данные какими они были до внесения изменений:



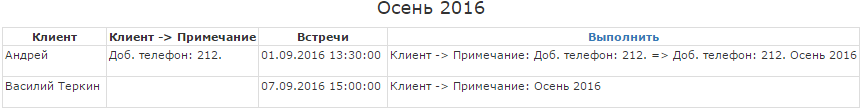
Но в таблице Клиентов мы увидим обновленные данные:



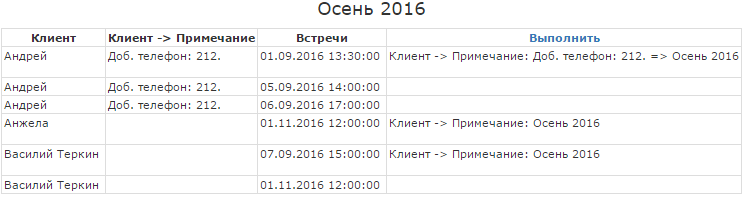
Обратите внимание, что в построителе отчетов присваиваемое значение заключено в кавычки, в отличие от полей Значение (от и до) и Функция (от и до). Поле «Присвоить» может использовать значения поля «Формула», а также функции языка SQL для вычисления присваиваемого значения. Для демонстрации этих возможностей на примере усложним задачу – будем не заменять Примечание Клиента, а дописывать к нему текст в конце с указанием номера телефона. Доработаем наш отчет следующим образом:



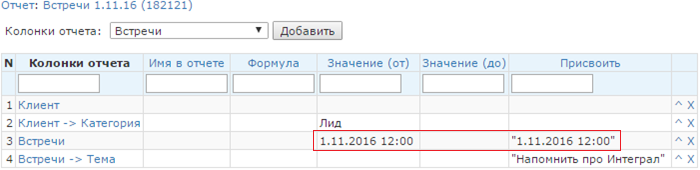
Теперь мы не будем терять информацию из Примечания, а просто дополним её:



У вас может возникнуть вопрос: что будет, если какой-то объект встречается в отчете больше одного раза, и мы пытаемся его изменить. Ответ: объект будет изменен только 1 раз. Например, в следующем отчете мы добавляем Примечание клиента, который упомянут в нескольких строках отчета, и Интеграл выполнит изменение только 1 раз для каждого клиента:

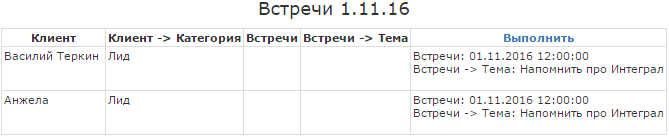


Теперь продемонстрируем создание новых записей. Пример: нам нужно запланировать на 1 ноября 2016 года встречи со всеми клиентами в статусе «Лид». Создадим такой отчет:

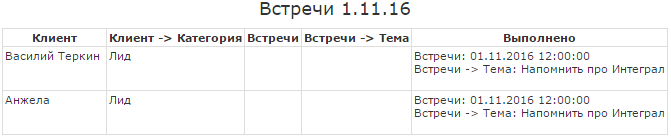


**Внимание! Если в колонке отчета заданы условия отбора и заполнено поле Присвоить, то Интеграл трактует условие отбора как *заданное условием или пустое*. Таким образом можно убедиться, что будет создано только одно новое значение и предотвратить создание дубликатов при многократном запуске отчета.**

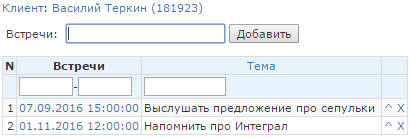
Этот отчет выдаст нам следующее:



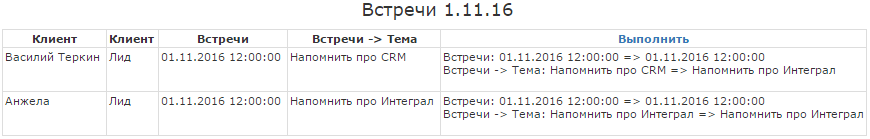
Как следует из отчета, у этих клиентов пока нет встреч на 1.11.2015, но Интеграл добавит им соответствующие записи, когда мы нажмем «Выполнить»:



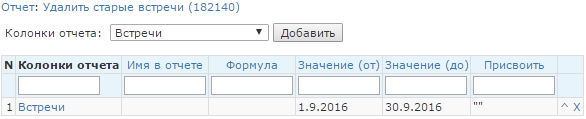
Зайдём во Встречи одного из этих клиентов и убедимся, что нужная запись создана:



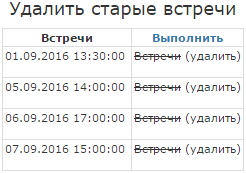
Если мы запустим этот отчет снова, то он уже не добавит еще раз эти записи, но исправит Тему встречи, если она будет отличаться от заданной (например, мы заменили Интеграл на CRM):



В следующем примере мы удалим информацию обо всех встречах сентября 2016 года:



Для удаления записей используется пустое значение: "". Интеграл не тратит память на пустые значения, поэтому информация будет просто удалена:



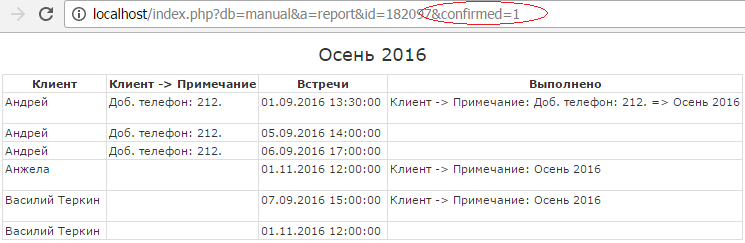
При нажатии кнопки «Выполнить» все эти записи будут удалены, включая все подчиненные им реквизиты (в том числе массивы).

Внимание! При удалении записей таким образом Интеграл не проводит проверку существующих связей, то есть, вы можете удалить даже справочные значения, использующиеся в других объектах (например, удалить менеджера, на которого есть ссылка у одного или более клиентов).

Рассмотренные примеры достаточно просты и затрагивают всего пару связанных таблиц. В реальных задачах вы можете использовать любое количество связанных таблиц и обновлять данные в любых полях этих таблиц одним запросом.

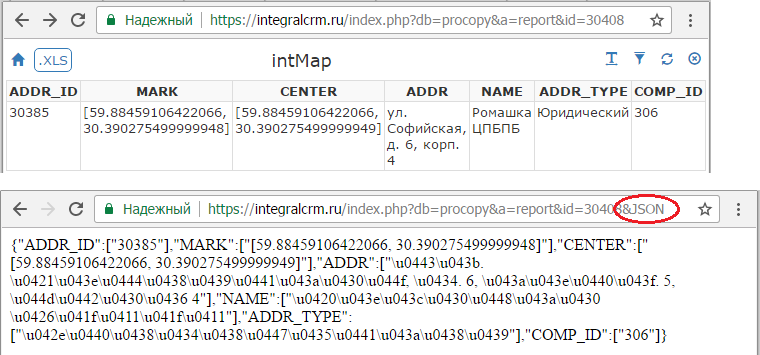
**Запросы на изменение Интеграла – это очень мощный инструмент, не имеющий аналогов в других системах хранения и управления данными.**

В строке адреса отчета, выполняющего изменения, можно наблюдать флаг «confirmed» – этот флаг сообщает Интегралу, что изменения подтверждены пользователем и должны быть немедленно выполнены. Вы можете использовать это знание для генерации ссылок на страницах интерфейса пользователя, чтобы выполнять изменения без подтверждения: по ссылке будет запущен отчет, который сразу выполнит запрошенные изменения.



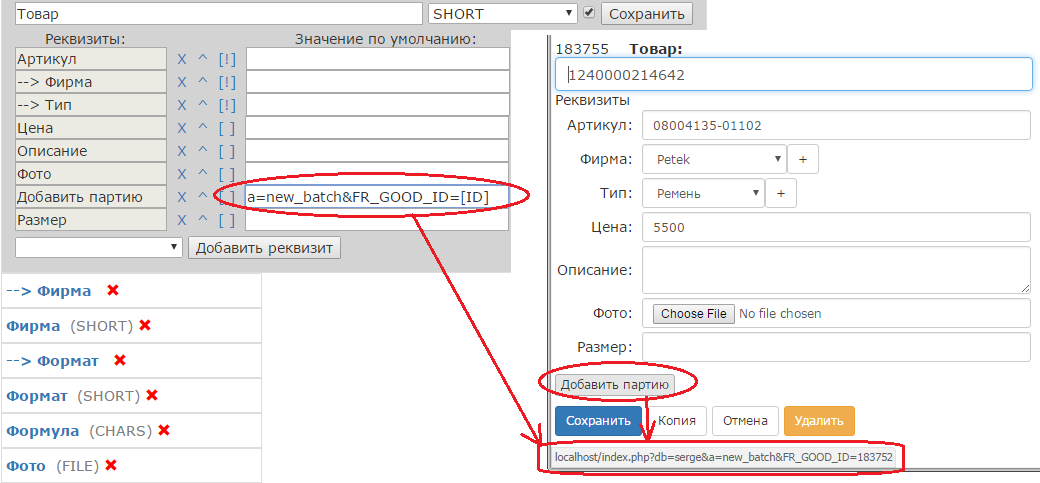
## Выгрузка отчета в формат JSON

Для построения интерфейсов и выгрузки данных может оказаться полезным выгрузка результатов отчета в виде JSON-файла. Для этого при запуске отчета следует передать параметр JSON, как показано на следующем рисунке:



## Реквизит типа Кнопка (BUTTON)

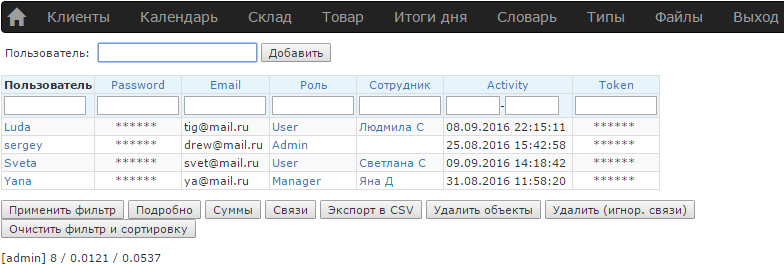
Среди [базовых типов](#Базовые_типы) данных есть особый тип – BUTTON. Реквизит этого типа отображается на форме редактирования объекта в виде «кнопки» – это некое действие с редактируемым объектом. Эта кнопка вызывает пользовательскую [html–форму](#_Пользовательские_html–формы) и передает ей в качестве параметра идентификатор текущего объекта. Для пользовательской формы можно настроить отчет, получив тем самым любое рабочее место для операций над текущим объектом и любыми связанными данными.



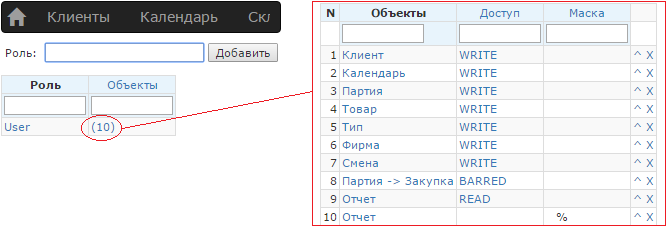
Таким образом, вы можете сверстать [в html](#_Пользовательские_html–формы) любое рабочее место и передать в него данные конкретного объекта для выполнения нужных вам действий.

## Разграничение прав пользователей

Интеграл поддерживает разграничение прав пользователей при помощи ролей доступа с детализацией до отдельных полей и их значений. Список пользователей находится в меню Словарь –> Пользователь. Имя пользователя, работающего в системе в данный момент, отображено в нижнем левом углу каждой страницы (см. рисунок ниже). Для каждого пользователя задана Роль:

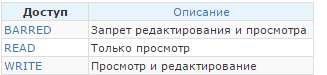


Кликнув название роли пользователя, можно просмотреть детали этой роли и заданные ей объекты:

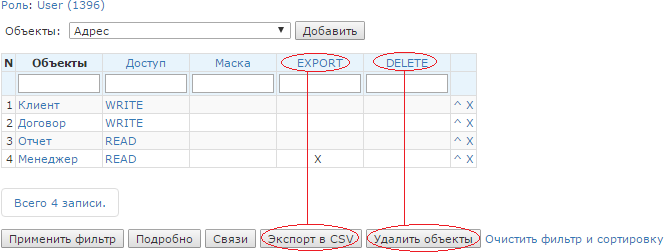


На рисунке показаны все разрешенные для просмотра и редактирования типы объектов, а также видно, что колонка «Закупка» (цена закупки товара) в таблице партия будет скрыта от пользователя с ролью User. Кроме того, отчеты, доступные только для чтения, ограничены маской « %», то есть пользователь увидит только те отчеты, название которых начинается с двух пробелов.

Уровни доступа Интеграла заданы в справочнике Словарь –> Доступ:



На уровне Роли Интеграл позволяет разрешить или запретить доступ пользователей к функциям экспорта и массового удаления данных определенный типов:

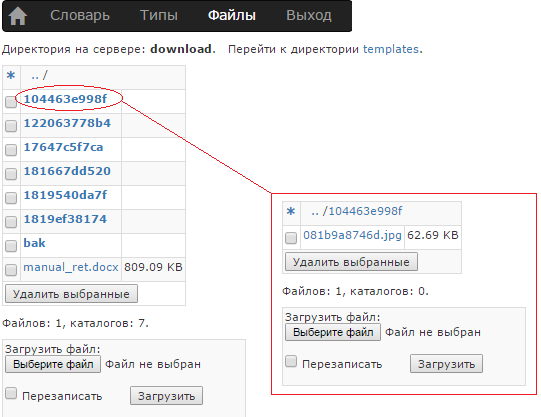


## Пользовательские файлы

**Меню Файлы в интеграле доступны только пользователю с именем, совпадающим с именем базы, и пользователю с именем admin.**

Все файлы, загруженные в систему пользователями через интерфейс Интеграла (например, фотографии товаров или файлы договоров), сохраняются в директории *download*, доступной администратору Интеграла. Для исключения неавторизованного доступа файлы хранятся в папках с зашифрованными именами, не более 1000 файлов в каждой папке. Имена зашифрованы с использованием имени базы, поэтому у каждой базы получается свой алгоритм шифрования.

У каждого файла есть системный идентификатор, из которого вычисляется имя папки и имя файла: имя файла – это последние 3 цифры идентификатора + шифр из 8 цифр; имя папки – это остальные цифры идентификатора + шифр из 8 цифр. На следующем рисунке показано, как будет храниться файл с идентификатором 104081 и расширением jpg:



Зная идентификатор файла, вы можете визуально найти его среди остальных. Вы можете скачивать файлы на свой компьютер, а также загружать их на сервер, при необходимости перезаписывая существующие файлы.

Обратите внимание, что файлы сохраняются в зашифрованных папках только будучи загружены через форму редактирования объекта – edit\_obj (см. [формы](#_Архитектура_приложения_Интеграл)), то есть при нормальной работе пользователей Интеграла). Если администратор загрузит файл через меню *Файлы*, то он будет загружен как есть. Поэтому если вы загружаете в эту папку актуальные версии руководства пользователя и/или служебных инструкций, то рекомендуется использовать секретное (хешированное) имя файла.

## Генерация web–страниц Интеграла

Базовый интерфейс Интеграла построен на пяти универсальных [формах](#_Архитектура_приложения_Интеграл) по единому принципу: html–шаблон содержит точки вставки информации из системных отчетов. Все эти формы, кроме отчета, вставляются в главную форму *main.html*, шаблон которой доступен для редактирования администратору. Администратор также может создавать свои html–формы и использовать их как внутри главной формы, так и самостоятельно.

На каждое действие пользователя генерируется страница рабочего места с одной из упомянутых универсальных форм (или пользовательской формой). Пользователю нет необходимости обращать внимание на адресную строку браузера, но аналитику Интеграла нужно знать несколько простых правил для внедрения в систему собственных html–форм.

Адрес страницы выглядит примерно так:

[https://integralcrm.ru/index.php?**db**=ret&**id**=776&**a**=object](https://integralcrm.ru/index.php?db=ret&id=776&a=object),

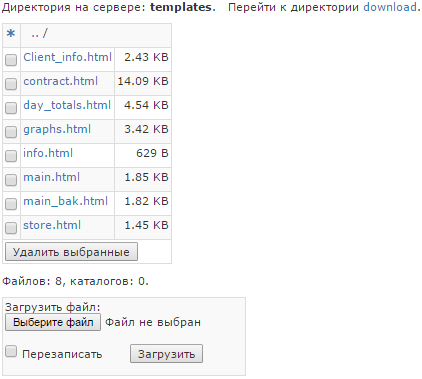
где синим цветом выделены параметры:

* db – имя базы данных (ret);
* a – имя универсальной формы (object);
* id – идентификатор объекта, над которым производится действие (776).

Задача аналитика при разработке html–формы: правильно подготовить и передать параметры для ее отображения. На данном этапе дорабатывается спецификация для более простого и быстрого создания собственного функционала, и вскоре она будет опубликована.

## Пользовательские html–формы и блоки данных

Все пользовательские html–формы находятся в меню *Файлы*, в директории *templates*:



В коде форм встречаются объявления блоков данных – это кубики, из которых собирается Интеграл:

<!-- Begin: Main --> … <!-- End: Main -->, <!-- Begin: Store --> … <!-- End: Store --> и <!-- File: a -->

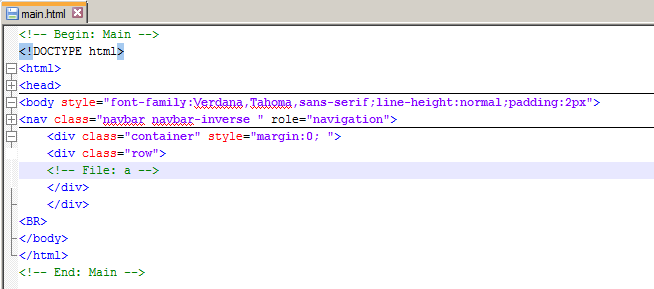
Эти конструкции обозначают блоки для вставки информации из отчетов. При построении страницы Интеграл заполняет точки вставки, заключенные в фигурные скобки {...} в блоках данных, пока у него не закончатся данные.

Есть две полезные конструкции для обращения к значениям родительского блока и значениям, переданным запросом POST или GET:

* Встретив выражение {\_parent\_.GOOD\_ID}, Интеграл будет просматривать все родительские блоки, пока не обнаружит первое значение GOOD\_ID.
* Конструкция {\_request\_.GOOD\_ID} проверит наличие параметра GOOD\_ID, переданного с запросом, и будет использовать его значение, если таковое задано.

**Блоки данных – третий важнейший элемент, помимо отчетов и типов. Принцип работы Интеграла очень прост: берем шаблон, смотрим, какие там блоки, ищем соответствующие отчеты и заполняем блоки данными.**

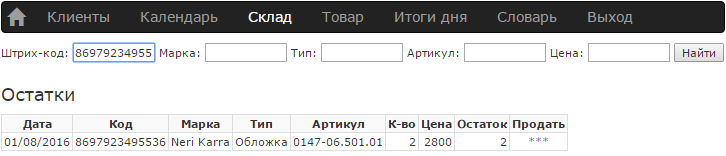
Некоторые отчеты являются служебными, например, для блока Main в главной форме Интеграла, представленной на следующем рисунке.



Конструкция <!-- File: a --> означает, что будет найден файл, имя которого содержится в переменной a, переданной с запросом POST или GET, и этот файл будет вставлен вместо этой конструкции. При этом в этом файле могут содержаться точки вставки данных и другие файлы, что будет также обработано Интегралом и заполнено данными.

**Внимание! Имя файла должно передаваться без указания расширения html.**

Рассмотрим пример создания пользовательской формы для оперативного просмотра остатков на складе (пользовательское меню Склад). Форма склада будет вложена в главную страницу системы main.html при помощи только что рассмотренной конструкции File. На этой форме пользователь может искать товар по нескольким параметрам, по умолчанию – по штрих–коду:

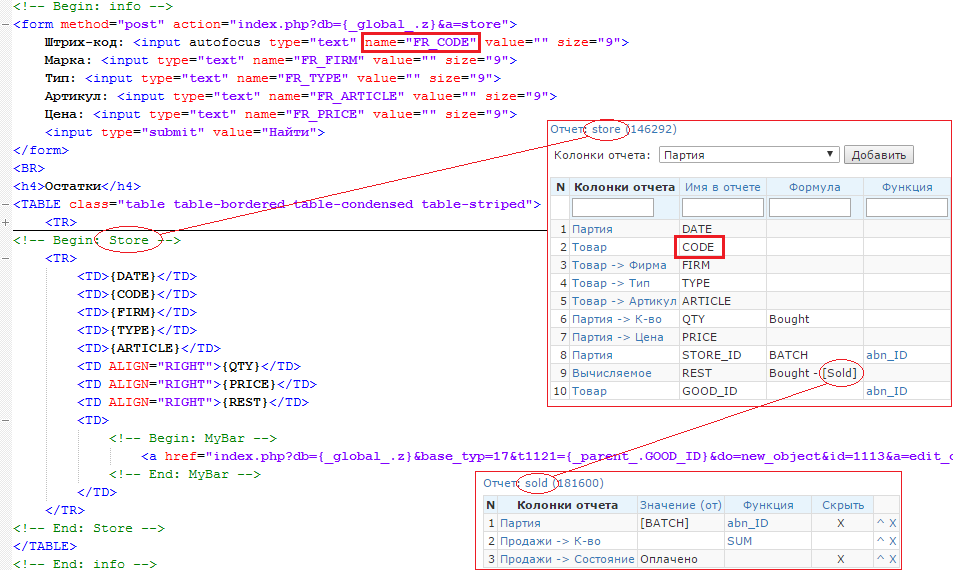


Файл формы Склада в директории templates – store.html, а его код представлен на следующем рисунке.

Форма состоит, в частности, из элемента html <form> – форма, которая вызывает сама себя (см. *action="index.php?db={\_global\_.z}&a=store"*) и передает введенные параметры: Штрих-код (FR\_CODE), Марка (FR\_FIRM) и другие. Эти параметры будут использованы в отчете, как описано в разделе [Вызов отчета с произвольными параметрами](#_Вызов_отчета_с).

Блок *Main* является служебным блоком Интеграла, в который заключены все html–тексты, из которых строятся страницы. Интеграл заполнит его сам по своим внутренним отчетам.

А вот блок *Store* не является служебным, поэтому Интеграл попытается найти соответствующий отчет в справочнике отчетов, выполнить его, а из его колонок заполнить точки вставки в блоке Store: {DATE}, {FIRM} и остальные. Сколько строк вернет отчет Store, столько раз повторится на странице кусок html–кода между тегами <!-- Begin: Store --> и <!-- End: Store -->, а значения из отчета попадут в соответствующие точки вставки.



По умолчанию при открытии этой формы фокус попадет в поле штрих–код (оно помечено «autofocus»), и если при этом пользователь введет код и нажмет Enter или отсканирует код сканером, то форма вызовет сама себя с параметром FR\_CODE, который применится в виде фильтра по полю CODE в отчете Store (см. выделенное красными прямоугольниками на рисунке).

На приведенном выше рисунке меню Склад был отсканирован код 8697923495536, для которого отчет вернул одну запись, и эта запись была в итоге отражена в таблице на форме Склада.

Несколько моментов, на которые нужно обратить внимание:

* блоки, обозначенные зеленым цветом в html–коде, являются вложенными: *info* содержит *Store*, который, в свою очередь, содержит блок *MyBar*;
* из блока *MyBar* мы можем обратиться к вышестоящему блоку и запросить у него данные – идентификатор товара GOOD\_ID, для чего используем {\_parent\_.GOOD\_ID};
* при отправке формы на сервер интеграла следует использовать адрес index.php?db={\_global\_.z}&a=store, где a – имя вызываемой формы (в данном случае store);
* {\_global\_.z} – внутренняя [глобальная константа](#Константы) Интеграла, содержащая имя базы;
* блок *MyBar* – это гиперссылка не некое действие (продажа товара из этой конкретной строки), это действие создает объект «продажа», для чего содержит команду do=new\_object;
* вы можете сообщить Интегралу, какую форму открыть после того, как он выполнит заданное вами действие (например, создание «продажи», запущенное из блока *MyBar*);
* создавая произвольные формы (рабочие места) и направляя пользователя от одной из них к другой, вы можете моделировать бизнес–процессы любой сложности.

**Блоки данных и правила их заполнения дополняют интеграл до законченного средства разработки: с помощью этих трех инструментов (**[**редактор типов**](#_Редактор_типов_Интеграла)**,** [**построитель отчетов**](#_Отчеты) **и** [**пользовательские формы**](#_Пользовательские_html–формы_и)**) вы можете собрать приложение, ничем не ограниченное по функционалу и сложности модели данных.**

Здесь можно сделать всё, что доступно в любом языке программирования, при этом, как и было обещано, вам не придется программировать: вы никогда не познаете, например, что такое полиморфизм, наследование и инкапсуляция, что, несомненно, сбережет вам время и нервы.

Если вы внимательно прочитали всё до этого момента, то вы уже можете программировать в Интеграле всё, что угодно. Не бойтесь, начните с малого и увидите, как быстро втянетесь! Если возникнут вопросы, то мы вам обязательно поможем, обращайтесь: [support@bkint.ru](mailto:support@bkint.ru).

## Служебные слова, функции и константы

Интеграл содержит небольшое количество служебных слов и функций, которые дают контроль над происходящим в системе. В фильтрах таблиц и отчетов могут быть использованы следующие слова:

* [TODAY] – сегодняшний день, например, 15, если сейчас 15.09.2016 г.
* [NOW] – текущие дата и время
* [YESTERDAY] – вчерашний день, например, 31.10.2016, если сейчас 1.11.2016 г.
* [TOMORROW] – завтрашний день.
* [MONTH\_AGO] – день месяц назад, например, 28.02.2016, если сейчас 28.03.2016 г.
* [USER] – имя пользователя, вошедшего в систему (его также видно в левом нижнем углу любой формы системы)
* ! – префикс отрицания последующего условия, например, !% означает «пусто»

В построителе отчетов могут использоваться внутренние функции Интеграла (их немного и все они начинаются с префикса «abn\_»). Эти функции нужны аналитику для программирования собственного функционала в Интеграле, такого как формы для быстрого сохранения, контекстные действия и, вообще, любые операции над данными в интеграле:

* abn\_DATE2STR – преобразование даты в строку
* abn\_ID – внутренний идентификатор объекта
* abn\_NUM2STR – преобразование числа в строку
* abn\_REQ – внутренний идентификатор типа реквизита
* abn\_TYP – внутренний идентификатор типа объекта
* abn\_ORD – внутренний идентификатор порядка (номера) объекта среди равных ему
* abn\_UP – внутренний идентификатор родителя объекта
* abn\_ROWNUM – номер строки в отчете, применим только к вычисляемому полю

Полный список функций находится в меню Словарь –> Функция.

Также Интеграл поддерживает использование следующих глобальных констант:

* {\_global\_.z} – имя текущей базы данных;
* {\_global\_.USER} – имя пользователя, который вошел в систему;
* {\_global\_.USER\_ID} – идентификатор пользователя, который вошел в систему.

## Known issues

Интеграл постоянно развивается, поэтому в нем неизбежно появляются разного рода дефекты. Если вы столкнулись с какой-либо ошибкой или неудобством, посмотрите, может, это есть в списке ниже. Будем благодарны за сообщение о незарегистрированных проблемах на [support@bkint.ru](mailto:support@bkint.ru).

|  |  |
| --- | --- |
| **Дефект / Проблема** | **Как обойти** |
| Не работает фильтр в шапке отчета | Это возможно, если имя колонки не задано. Чтобы фильтр работал, следует задать имя колонки в отчете. Планируется к исправлению. |
| .xls выгружается без имени и не открывается в Android |  |
| При загрузке файла BKI часть данных не заходит | В .bki сейчас есть дефект с разрешением подчиненности при выгрузке и загрузке данных. Планируется к исправлению. |
|  |  |